GOAT

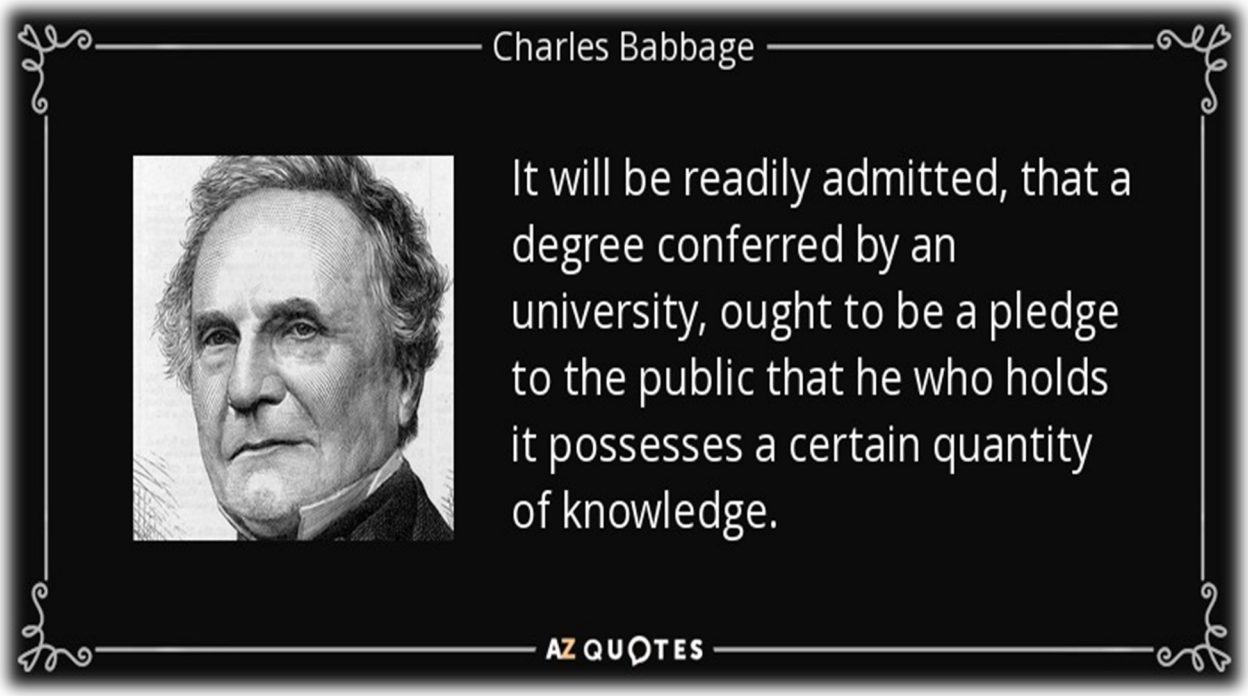
# 108
# Greatest Of All Times

GLOBALLY SELECTED
PERSONALITIES

ISBN:978-81-984229-8-9

Compiled by:
Prof Dr S Ramalingam

26 Dec 1791 <::><::><::> 18 Oct1871




Na Subbureddiar 100 Educational Trust
[An ISO 9001 - 2015 Certified]
AD-13,5th Street, Anna Nagar West, Chennai - 600 040
www.nasubbureddiar100.in

26 Dec 1791

18 Oct 1871

College of Science and Engineering & University Libraries

# CHARLES BABBAGE INSTITUTE

https://cse.umn.edu/cbi/who-was-charles-babbage

## Who was Charles Babbage?

### Manuscript Materials and Exhibits

- CBI holds a microfilmed copy of the Papers of Charles Babbage. The original papers are in the British Library.
- The University of Auckland, New Zealand, also has some of Babbage's materials. CBI has photocopies of their holdings and the inventory to these copies, as well as other Charles Babbage materials held in the Charles Babbage Collection (CBI 54)
- The Science Museum in London constructed Babbage's Difference Engine No. 2 in 1991. The Science Museum Library and Archives in Wroughton holds the most comprehensive set of original manuscripts and design drawings.

## Charles Babbage's Published Works

- A Comparative View of the Various Institutions for the Assurance of Lives (1826)
- Table of Logarithms of the Natural Numbers from 1 to 108,000 (1827)
- Reflections on the Decline of Science in England (1830)
- On the Economy of Machinery and Manufactures (1832)
- Ninth Bridgewater Treatise (1837)
- Passages from the Life of a Philosopher (1864)

## Publication About Charles Babbage


- Charles Babbage. *Passages from the Life of a Philosopher*. (New Brunswick, NJ: Rutgers University Press, Piscataway, NJ, 1994)
- Babbage, Henry Prevost . *Babbage's Calculating Engines: A Collection of Papers*. (Los Angeles: Tomash, 1982) Charles Babbage Institute Reprint Series, vol. 2.
- Bromley, Allan G. "The Evolution of Babbage's Calculating Engines" *Annals of the History of Computing*, 9 (1987): 113-136.
- Buxton, H. W. *Memoir of the Life and Labours of the Late Charles Babbage Esq., F.R.S.* (Cambridge, MA.: MIT Press, 1988) Charles Babbage Institute Reprint Series, vol. 13.
- Cambell-Kelly, Martin (ed.) *The Works of Charles Babbage* (11 vols.) (New York: New York University Press, 1989)
- Dubbey, John Michael. *Mathematical Work of Charles Babbage*. (Cambridge, MA: Cambridge University Press, 1978)
- Hyman, Anthony. *Charles Babbage: Pioneer of the Computer*. (Princeton, NJ: Princeton University Press, 1983)
- Moseley, Maboth. *Irascible Genius: a Life of Charles Babbage, Inventor*. (London: Hutchinson, 1964)
- Randell, Brian. "From Analytical Engine to Electronic Digital Computer: The Contributions of Ludgate, Torres, and Bush" *Annals of the History of Computing*, 4 (October 1982): 327.
- Swade, Doron. *Charles Babbage and his Calculating Engines*. (London: Science Museum, 1991)
- Swade, Doron. *The Cogwheel Brain: Charles Babbage and the Quest to build the first Computer*. (London: Little, Brown, 2001)

- Van Sinderen, Alfred W. "The Printed Papers of Charles Babbage" *Annals of the History of Computing*, 2 (April 1980); 169-185.


Publications on Babbage and Ada Lovelace


- Huskey, Velma R., and Harry D. Huskey. "Lady Lovelace and Charles Babbage" *Annals of the History of Computing* 2 (October 1980): 299-329.
- Stein, Dorothy. *Ada: A Life and A Legacy*. (Cambridge: MIT Press, 1985)
- Toole, B.A. "Ada Byron, Lady Lovelace, an analyst and metaphysician." Annals of the History of Computing 18 #3 (Fall 1996): 4-12.
- Fuegi, John, and Jo Francis. "Lovelace & Babbage and the creation of the 1843 'notes'." Annals of the History of Computing 25 #4 (Oct-Dec 2003): 16-26.
- Wikipedia entry on Ada Lovelace.


(😊)(😊)(😊)(😊)(😊)

**CHARLES BABBAGE:**
His Life and Contributions

https://cs.stanford.edu/people/eroberts/courses/soco/projects/1998-99/babbage/bio.htm

Technology today, in these last few years of the twentieth century, is growing so fast it is often hard to keep up with. The computer industry is booming, advancing with every passing year. The industry is still young, but its products are in such high demand that rapid advancement of technology is hardly surprising. What is surprising, however, is that it took the industry a second invention and over one hundred years before the power of computers was finally recognized. Though computers weren't ever built until the mid-1900s, the first idea for a computer was actually developed starting in 1837. Charles Babbage, born to a wealthy London family in 1791, was the brain behind the idea, and is famous for his work developing plans for two different computers. His first, the Difference Engine, was partially completed in the early 1830s, but never to the extent he had planned. The Analytical Engine, his second and more complex design was never completed at all, but both had the potential to be very powerful, especially for the time period.

His machines were essentially the first computers in history. Though a crude calculator was built by mathematician Blaise Pascal in the 1642, it was very unreliable, and had far fewer capabilities than even Babbage's Difference Engine. The Difference Engine could compute simple calculations, like multiplication or addition, but its most important trait was its ability create tables of the results of up to seven-degree polynomial functions. His engine, except for the printing mechanism, was close to being finished in 1832, but funds to complete the project were dropped, and so had to be his project. By 1837, Babbage had come up with a new idea: a computer that could understand commands and could be programmed much like a modern-day computer. He called it the Analytical Engine, and it was the first machine ever designed with the idea of programming. Babbage started working on this engine when work on his Difference Engine was halted and continued for most of his life. Unfortunately, due mostly to political and economic concerns, and of course to the high technological involvement of the project, Babbage's Analytical Engine was never completed. His ideas were isolated at the time, and no other attempts at building such a computer were thought of again until well into the next century. Babbage, a true computer pioneer, is known as the "Uncle" of computers, due to his early, but isolated contributions to the field.

Charles Babbage was an exceptional man. Obviously very intelligent, his mathematical and mechanical genius was apparent even at an early age. As a child, he liked to take apart toys in order to figure out how they worked. Later, in school, he learned algebra on his own because he was fascinated by the subject. His interest in mathematics continued into high school, where, because of the time constraints of a heavy, prep-school course load, Babbage would go to school in secret from three until five in the morning to study calculus by himself. By the time he went to university in 1810 he had a good understanding of and a hugely motivated interest in the field of mathematics.

Babbage went to Trinity College in Cambridge in 1810 to start his first year of university. While he was there, he encountered some fellow mathematicians, George Peacock and John Herschel, with whom he could study calculus. During the time, the main focus of higher education was the humanities, so most of their calculus studies had to be conducted on their own. Babbage and his friends, called the Analytical Society, found that the French had the best understanding of calculus and that their texts were far superior to those used in the English system, and they decided to translate them into English. One of the results of the Analytical Society's work was the popularization of the **dy/dx** (from the previous **x'**) notation now commonly used in differential calculus.

Through his studies at Cambridge and his work with the Analytical Society, Babbage became a well-respected mathematical scholar. At the age of twenty-five he became a member of the Royal Society, and was later awarded the title of Lucasian Professor of Mathematics, a title once held by Sir Isaac Newton

and now by physicist Stephen Hawking. Babbage's contributions to the field of mathematics made him very deserving of the award.

Babbage's love for mathematics carried through all aspects of his life. He loved numbers and orders and he was a collector of facts of any kinds. In fact, his collection was so involved that he would stop the check and record the pulse of every animal he came across. He believed that there was some perfect order in the universe, and that if one could just gather enough facts and compile them into tables, a careful analysis of them all would lead to some great universal understanding. Babbage took his Newtonian ideas into the real world, and tried to win some money using them at the horse races. By calculating and organizing as many facts about as many horses as he could, he practiced his powers of prediction, which unfortunately did not measure up to his expectations.

Babbage was also fascinated by the supernatural world. He was interested in religion from a scientific standpoint, and believed God to be the ultimate programmer. He was curious about the darker side of religion as well, forming ghost-hunting organizations in college and harboring a fascination with the Devil. On one adolescent occasion, he went upstairs into his attic, drew blood from his fingers, and, with the blood, painted a circle on the floor just large enough for him to stand inside. Stepping into his circle, he chanted the Lord's Prayer backwards and watched for a sign: a raven, a bat, a ghost, anything to let him know that the Devil was there. Nothing came to him, but he continued to try little tests in order to gain some understanding of the supernatural world. While in college, he made a promise with his friend that the first one who died had to visit the other while he was still living. His friend eventually died before Babbage, and again Babbage looked for some kind of sign. Babbage never found what he was looking for, and before long, his ideas about religion went from traditional to purely scientific.

Charles Babbage was an intellectual man, respected by almost everyone in his mathematical field; however, he had a strange personality to which some people had difficulty relating. Part of the reason his work on the first Difference Engine was stopped was because of a fight he got into with one of his engineers, Joseph Clement. Babbage was advised to check up on the work Clement was doing to make sure he was paying the engineer the correct amount. Clement was reluctant, but agreed to the checks. Eventually the process destroyed the trust between the two men, and their collaboration was halted in 1832, as was the funding for the project.

Babbage also got into political issues during the Analytical Engine-building process. Since these machines were so big and required so much material, not to mention the time Babbage devoted to them, the planning and construction of his engines, especially the Analytical Engine, was costly. Babbage applied for grants from the government to fund his project, because, after all, the government would probably get the most use out of the finished product. He

was awarded some money, but not nearly enough to complete his work. After a while, when Babbage couldn't produce much of a product for the government, they started growing skeptical of his ability to finish the machine. Also working against him was one of Babbage's colleagues who contiguously urged the government not to fund the project, calling it "worthless." Interestingly enough, this same mathematician had tried to create a similar machine (to the Difference Engine) himself and had failed.

Babbage eventually stopped most of his work on his engines by 1848. In 1848, he designed the Difference Engine number two, based upon the improvements in time and space that had come about while thinking of the Analytical Engine. This model was later to be built in 1991 by the London Science Museum. In 1854, a Swedish printing press owner, George Scheutz completed a difference engine, complete with a printing mechanism, based upon a design of Babbage's he had acquired in 1834. Babbage was impressed by the work and recommended that Scheutz win a medal from the Royal Society. No version of an analytical engine was ever completed. Despite the failure to realize his ultimate dream, Babbage's plans alone distinguish himself as a man ahead of the times. He was almost too ahead of his times, as his plans didn't spark any more development in the field, but the innovations are nonetheless remarkable, and stand as an inspiration to computer visionaries today.

(:))@@@@@@@@@@@@@@@@@@@@(😊)

# Computing Pioneer

**[https://en.wikipedia.org/wiki/Charles_Babbage]**

Part of Charles Babbage's Difference Engine (#1), assembled after his death by his son, Henry Prevost Babbage (1824–1918), using parts found in Charles' laboratory. Whipple Museum of the History of Science, Cambridge, England.

Babbage's machines were among the first mechanical computers. That they were not actually completed was largely because of funding problems and clashes of personality, most notably with George Biddell Airy, the Astronomer Royal.

Babbage directed the building of some steam-powered machines that achieved some modest success, suggesting that calculations could be mechanised. For more than ten years he received government funding for his project, which amounted to £17,000, but eventually the Treasury lost confidence in him.

While Babbage's machines were mechanical and unwieldy, their basic architecture was similar to that of a modern computer. The data and program memory were separated, operation was instruction-based, the control unit could make conditional jumps, and the machine had a separate I/O unit.

## Background on mathematical tables

In Babbage's time, printed mathematical tables were calculated by human computers; in other words, by hand. They were central to navigation, science and engineering, as well as mathematics. Mistakes were known to occur in transcription as well as calculation.

At Cambridge, Babbage saw the fallibility of this process, and the opportunity of adding mechanisation into its management. His own account of his path towards mechanical computation references a particular occasion:

In 1812 he was sitting in his rooms in the Analytical Society looking at a table of logarithms, which he knew to be full of mistakes, when the idea occurred to him of computing all tabular functions by machinery. The French government had produced several tables by a new method. Three or four of their mathematicians decided how to compute the tables, half a dozen more broke down the operations into simple stages, and the work itself, which was restricted to addition and subtraction, was done by eighty computers who knew only these two arithmetical processes. Here, for the first time, mass production was applied to arithmetic, and Babbage was seized by the idea that the labours of the unskilled computers [people] could be taken over completely by machinery which would be quicker and more reliable.

There was another period, seven years later, when his interest was aroused by the issues around computation of mathematical tables. The French official initiative by Gaspard de Prony, and its problems of implementation, were familiar to him. After the Napoleonic Wars came to a close, scientific contacts were renewed on the level of personal contact: in 1819 Charles Blagden was in Paris looking into the printing of the stalled de Prony project, and lobbying for the support of the Royal Society. In works of the 1820s and 1830s, Babbage referred in detail to de Prony's project.

## Difference engine

**The Science Museum's Difference Engine No. 2, built from Babbage's design**

Portion of Babbage's difference engine

Babbage began in 1822 with what he called the difference engine, made to compute values of polynomial functions. It was created to calculate a series of values automatically. By using the method of finite differences, it was possible to avoid the need for multiplication and division.

For a prototype difference engine, Babbage brought in Joseph Clement to implement the design, in 1823. Clement worked to high standards, but his machine tools were particularly elaborate. Under the standard terms of business of the time, he could charge for their construction, and would also own them. He and Babbage fell out over costs around 1831.

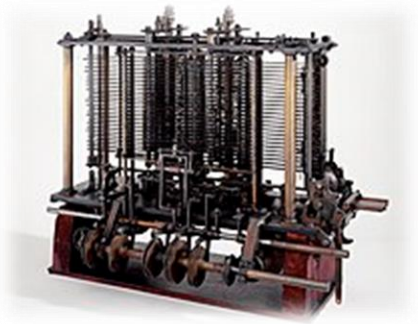
Some parts of the prototype survive in the Museum of the History of Science, Oxford. This prototype evolved into the "first difference engine". It remained unfinished and the finished portion is located at the Science Museum in London. This first difference engine would have been composed of around 25,000 parts, weighed fifteen short tons (13,600 kg), and would have been 8 ft (2.4 m) tall. Although Babbage received ample funding for the project, it was never completed. He later (1847–1849) produced detailed drawings for an improved version,"Difference Engine No. 2", but did not receive funding from the British government. His design was finally constructed in 1989–1991, using his plans and 19th-century manufacturing tolerances. It performed its first calculation at the Science Museum, London, returning results to 31 digits.

Nine years later, in 2000, the Science Museum completed the printer Babbage had designed for the difference engine.

## Completed models

The Science Museum has constructed two Difference Engines according to Babbage's plans for the Difference Engine No 2. One is owned by the museum. The other, owned by the technology multimillionaire Nathan Myhrvold, went on exhibition at the Computer History Museum in Mountain View, California on 10 May 2008. The two models that have been constructed are not replicas.

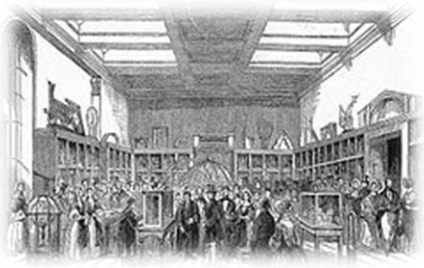
## Analytical Engine



**Portion of the mill with a printing mechanism of the Analytical Engine, built by Charles Babbage, as displayed at the Science Museum (London)**

After the attempt at making the first difference engine fell through, Babbage worked to design a more complex machine called the Analytical Engine. He hired C. G. Jarvis, who had previously worked for Clement as a draughtsman. The Analytical Engine marks the transition from mechanised arithmetic to fully-fledged general purpose computation. It is largely on it that Babbage's standing as computer pioneer rests.

The major innovation was that the Analytical Engine was to be programmed using punched cards: the Engine was intended to use loops of Jacquard's punched cards to control a mechanical calculator, which could use as input the results of preceding computations. The machine was also intended to employ several features subsequently used in modern computers, including sequential control, branching and looping. It would have been the first mechanical device to be, in principle, Turing-complete. Charles Babbage wrote a series of programs for the Analytical Engine from 1837 to 1840. The first program was finished in 1837. The Engine was not a single physical machine, but rather a succession of designs that Babbage tinkered with until his death in 1871.



**Part of the Analytical Engine on display, in 1843, left of centre in this engraving of the King George III Museum in King's College, London**

## Ada Lovelace and Italian followers

Ada Lovelace, who corresponded with Babbage during his development of the Analytical Engine, is credited with developing an algorithm that would enable the Engine to calculate a sequence of Bernoulli numbers. Despite documentary evidence in Lovelace's own handwriting, some scholars dispute to what extent the ideas were Lovelace's own. For this achievement, she is often described as the first computer programmer; though no programming language had yet been invented.

Lovelace also translated and wrote literature supporting the project. Describing the engine's programming by punch cards, she wrote: "We may say most aptly that the Analytical Engine weaves algebraical patterns just as the Jacquard loom weaves flowers and leaves."

Babbage visited Turin in 1840 at the invitation of Giovanni Plana, who had developed in 1831 an analog computing machine that served as a perpetual calendar. Here in 1840 in Turin, Babbage gave the only public explanation and lectures about the Analytical Engine. In 1842 Charles Wheatstone approached Lovelace to translate a paper of Luigi Menabrea, who had taken notes of Babbage's Turin talks; and Babbage asked her to add something of her own. Fortunato Prandi who acted as interpreter in Turin was an Italian exile and follower of Giuseppe Mazzini.

## Swedish followers

Per Georg Scheutz wrote about the difference engine in 1830, and experimented in automated computation. After 1834 and Lardner's *Edinburgh Review* article he set up a project of his own, doubting whether Babbage's initial plan could be carried out. This he pushed through with his son, Edvard Scheutz.[179] Another Swedish engine was that of Martin Wiberg (1860).[180]

## Legacy

In 2011, researchers in Britain proposed a multimillion-pound project, "Plan 28", to construct Babbage's Analytical Engine. Since Babbage's plans were continually being refined and were never completed, they intended to engage the public in the project and crowd-source the analysis of what should be built. It would have the equivalent of 675 bytes of memory, and run at a clock speed of about 7 Hz. They hoped to complete it by the 150th anniversary of Babbage's death, in 2021.

Advances in MEMS and nanotechnology have led to recent high-tech experiments in mechanical computation. The benefits suggested include operation in high radiation or high temperature environments. These modern versions of mechanical computation were highlighted in *The Economist* in its special "end of the millennium" black cover issue in an article entitled "Babbage's Last Laugh".

Due to his association with the town Babbage was chosen in 2007 to appear on the 5 Totnes pound note. An image of Babbage features in the British cultural icons section of the newly designed British passport in 2015.

(😊)@@@@@@@@@@@@@@@@@@@@(😊)

SCIENCE MUSEUM GROUP HODS **Charles Babbage, The Father of the Computer**

https://www.historyofdatascience.com/charles-babbage-2/

Charles Babbage (1791–1871) was an English mathematician and inventor. He is credited with designing the first digital automatic computer, which contained all the essential concepts found in the ones we use today.

Born in London, Charles Babbage studied at Trinity College Cambridge — although he had already taught himself many aspects of contemporary mathematics. It was during this time that he first had the idea of mechanically calculating mathematical tables. In 1823, he obtained government support to design a projected machine, the Difference Engine, with a 20-decimal capacity. Like modern computers, it could store data for later processing. Charles began developing the mechanical engineering techniques while serving as Lucasian Professor of Mathematics at the University of Cambridge. However, the full room-sized engine was never built as the metalworking techniques of the era were not precise enough and too costly.

"Errors using inadequate data are much less than those using no data at all."

## Brilliant Ideas Before Their Time

By the mid-1830s, Charles was already preparing plans for an improved and more complex design: the Analytical Engine, the precursor of the modern digital computer. He envisaged that it would be capable of performing any arithmetical operation based on instructions from punched cards, a memory unit to store numbers, sequential control, and many other basics found in present-day computers. The project was far more advanced than anything that had ever been built before — with a memory unit large enough to hold 1,000 50-digit numbers. It was intended to be steam-driven and run by one attendant. In 1843, Charles Babbage's friend mathematician Ada Lovelace published a paper explaining how the engine could perform a sequence of calculations. The first computer program was born.

The Analytical Engine, however, was never completed. The ambitious design was, once again, difficult to implement with the technology that existed in the 19th century. In 1991, British scientists built the Difference Engine No. 2 — accurate to 31 digits — to Charles' specifications. Their success indicates that his idea would have worked. In 2000, the printer for the Difference Engine was also built.

In addition to inventing early computer concepts, Charles Babbage also helped establish the modern postal system in England and compiled the first reliable actuarial tables. He invented a speedometer, as well as the train cow-catcher to deflect obstacles on the track.

## Key Dates

- **1812**
  **LEARNING FROM EUROPE**
  Charles Babbage helps found the Analytical Society to introduce mathematical developments from Europe to England.

  **1816**
  **EARLY RECOGNITION**
  Charles Babbage is elected a fellow of the Royal Society of London. He also plays a key role in founding the Royal Astronomical (1820) and Statistical (1834) Societies.

  **1832**
  **INDUSTRIAL INSIGHTS**
  Charles Babbage publishes "On the Economy of Machinery and Manufactures," which explores the organization of industrial production. The book sells well and goes to a fourth edition.

(😊)(😊)😊)(😊)(😊)

MacTutor

https://mathshistory.st-andrews.ac.uk/Biographies/Babbage/

# Charles Babbage

**Quick Info**

**Born**
26 December 1791
London, England
**Died**
18 October 1871
London, England

---


## Summary

**Charles Babbage** originated the modern analytic computer. He invented the principle of the analytical engine, the forerunner of the modern electronic computer.

View seven larger pictures

## Biography

Both the date and place of **Charles Babbage**'s birth were uncertain but have now been firmly established. In [1] and [12], for example, his date of birth is given as 26 December 1792 and both give the place of his birth as near Teignmouth. Also in [18] it is stated:-

*Little is known of Mr Babbage's parentage and early youth except that he was born on* 26 *December* 1792*.*

However, a nephew wrote to *The Times* a week after the obituary [18] appeared, saying that Babbage was born on 26 December 1791. There was little evidence to prove which was right until Hyman (see [8]) in 1975 found that Babbage's birth had been registered in St Mary's Newington, London on 6 January 1792. Babbage's father was Benjamin Babbage, a banker, and his mother was Betsy Plumleigh Babbage. Given the place that his birth was registered Hyman says in [8] that it is almost certain that Babbage was born in the family home of 44 Crosby Row, Walworth Road, London.

Babbage suffered ill health as a child, as he relates in [4]:-

*Having suffered in health at the age of five years, and again at that of ten by violent fevers, from which I was with difficulty saved, I was sent into Devonshire and placed under the care of a clergyman* (*who kept a school at Alphington, near Exeter*)*, with instructions to attend to my health; but, not to press too much knowledge upon me: a mission which he faithfully accomplished.*

Since his father was fairly wealthy, he could afford to have Babbage educated at private schools. After the school at Alphington he was sent to an academy at Forty Hill, Enfield, Middlesex where his education properly began. He began to show a passion for mathematics but a dislike for the classics. On leaving the academy, he continued to study at home, having an Oxford tutor to bring him up to university level. Babbage in [4] lists the mathematics books he studied in this period with the tutor:-

*Amongst these were Humphry Ditton's 'Fluxions', of which I could make nothing; Madame Agnesi's 'Analytical Instructions' from which I acquired some*

*knowledge; Woodhouse's 'Principles of Analytic Calculation', from which I learned the notation of Leibniz; and Lagrange's 'Théorie des Fonctions'. I possessed also the 'Fluxions' of Maclaurin and of Simson.*

Babbage entered Trinity College, Cambridge in October 1810. However the grounding he had acquired from the books he had studied made him dissatisfied with the teaching at Cambridge. He wrote [4]:-

*Thus it happened that when I went to Cambridge I could work out such questions as the very moderate amount of mathematics which I then possessed admitted, with equal facility, in the dots of Newton, the d's of Leibniz, or the dashes of Lagrange. I thus acquired a distaste for the routine of the studies of the place, and devoured the papers of Euler and other mathematicians scattered through innumerable volumes of the academies of St Petersburg, Berlin, and Paris, which the libraries I had recourse to contained.*

*Under these circumstances it was not surprising that I should perceive and be penetrated with the superior power of the notation of Leibniz.*

It is a little difficult to understand how Woodhouse's *Principles of Analytic Calculation* was such an excellent book from which to learn the methods of Leibniz, yet Woodhouse was teaching Newton's calculus at Cambridge without any reference to Leibniz's methods. Woodhouse was one of Babbage's teachers at Cambridge yet he seems to have taken no part in the Society that Babbage was to set up to try to bring the modern continental mathematics to Cambridge.

Babbage tried to buy Lacroix's book on the differential and integral calculus but this did not prove easy in this period of war with Napoleon. When he did find a copy of the work he had to pay seven guineas for it - an incredible amount of money in those days. Babbage then thought of setting up a Society to translate the work [4]:-

*I then drew up the sketch of a society to be instituted for translating the small work of Lacroix on the Differential and Integral Calculus. It proposed that we should have periodical meetings for the propagation of d's; and consigned to perdition all who supported the heresy of dots. It maintained that the work of Lacroix was so perfect that any comment was unnecessary.*

Babbage talked with his friend Edward Bromhead (who would become George Green's friend some years later- see the article on Green) who encouraged him to set up his Society. The Analytical Society was set up in 1812 and its members were all Cambridge undergraduates. Nine mathematicians attended the first meeting but the two most prominent members, in addition to Babbage, were John Herschel and George Peacock.

Babbage and Herschel produced the first of the publications of the Analytical Society

when they published *Memoirs of the Analytical Society* in 1813. This is a remarkably deep work when one realises that it was written by two undergraduates. They gave a history of the calculus, and of the Newton, Leibniz controversy they wrote:-

*It is a lamentable consideration, that that discovery which has most of any done honour to the genius of man, should nevertheless bring with it a train of reflections so little to the credit of his heart.*

Two further publications of the Analytical Society were the joint work of Babbage, Herschel and Peacock. These are the English translation of Lacroix's *Sur le calcul différentiel et intégral* published in 1816 and a book of examples on the calculus which they published in 1820.

Babbage had moved from Trinity College to Peterhouse and it was from that College that he graduated with a B.A. in 1814. However, Babbage realised that Herschel was a much more powerful mathematician than he was so [12]:-

*He did not compete for honours, believing Herschel sure of first place and not caring to come out second.*

Indeed Herschel was first Wrangler, Peacock coming second. Babbage married in 1814, then left Cambridge in 1815 to live in London. He wrote two major papers on functional equations in 1815 and 1816. Also in 1816, at the early age of 24, he was elected a fellow of the Royal Society of London. He wrote papers on several different mathematical topics over the next few years but none are particularly important and some, such as his work on infinite series, are clearly incorrect.

Babbage was unhappy with the way that the learned societies of that time were run. Although elected to the Royal Society, he was unhappy with it. He was to write of his feelings on how the Royal Society was run:-

*The Council of the Royal Society is a collection of men who elect each other to office and then dine together at the expense of this society to praise each other over wine and give each other medals.*

However in 1820 he was elected a fellow of the Royal Society of Edinburgh, and in the same year he was a major influence in founding the Royal Astronomical Society. He served as secretary to the Royal Astronomical Society for the first four years of its existence and later he served as vice-president of the Society.

Babbage, together with Herschel, conducted some experiments on magnetism in 1825, developing methods introduced by Arago. In 1827 Babbage became Lucasian Professor of Mathematics at Cambridge, a position he held for 12 years although he never taught. The reason why he held this prestigious post yet failed to carry out the duties one would have expected of the holder, was that by this time

he had become engrossed in what was to became the main passion of his life, namely the development of mechanical computers.

Babbage is without doubt the originator of the concepts behind the present day computer. The computation of logarithms had made him aware of the inaccuracy of human calculation around 1812. He wrote in [4]:-

*... I was sitting in the rooms of the Analytical Society, at Cambridge, my head leaning forward on the table in a kind of dreamy mood, with a table of logarithms lying open before me. Another member, coming into the room, and seeing me half asleep, called out, Well, Babbage, what are you dreaming about?" to which I replied "I am thinking that all these tables"* (*pointing to the logarithms*) *"might be calculated by machinery."*

Certainly Babbage did not follow up this idea at that time but in 1819, when his interests were turning towards astronomical instruments, his ideas became more precise and he formulated a plan to construct tables using the method of differences by mechanical means. Such a machine would be able to carry out complex operations using only the mechanism for addition. Babbage began to construct a small difference engine in 1819 and had completed it by 1822. He announced his invention in a paper *Note on the application of machinery to the computation of astronomical and mathematical tables* read to the Royal Astronomical Society on 14 June 1822.

Although Babbage envisaged a machine capable of printing out the results it obtained, this was not done by the time the paper was written. An assistant had to write down the results obtained. Babbage illustrated what his small engine was capable of doing by calculating successive terms of the sequence n2+n+41$n2+n+41$.

The terms of this sequence are 41, 43, 47, 53, 61, ... while the differences of the terms are 2, 4, 6, 8, .. and the second differences are 2, 2, 2, ..... The difference engine is given the initial data 2, 0, 41; it constructs the next row 2, (0 + 2), [41 + (0 + 2)], that is 2, 2, 43; then the row 2, (2 + 2), [43 + (2 + 2)], that is 2, 4, 47; then 2, 6, 53; then 2, 8, 61; ...

Babbage reports that his small difference engine was capable of producing the members of the sequence n2+n+41$n2+n+41$ at the rate of about 60 every 5 minutes.

Babbage was clearly strongly influenced by de Prony's major undertaking for the French Government of producing logarithmic and trigonometric tables with teams of people to carry out the calculations. He argued that a large difference engine could do the work undertaken by teams of people saving cost and being totally accurate.

On 13 July 1824 Babbage received a gold medal from the Astronomical Society for his development of the difference engine. He then met the Chancellor of the Exchequer to seek public funds for the construction of a large difference engine.

The Royal Society had already given positive advice to the government:-

*Mr Babbage has displayed great talent and ingenuity in the construction of his machine for computation, which the committee thanks fully adequate to the attainment of the objects proposed by the inventory; and they consider Mr Babbage as highly deserving of public encouragement, in the prosecution of his arduous undertaking.*

His initial grant was for £1500 and he began work on a large difference engine which he believed he could complete in three years. He set out to produce an engine with [3]:-

*... six orders of differences, each of twenty places of figures, whilst the first three columns would each have had half a dozen additional figures.*

Such an engine would easily have been able to compute all the tables that de Prony had been calculating, and it was intended to have a printer to print out the results automatically. However the construction proceeded slower than had been expected. By 1827 the expenses were getting out of hand.

The year 1827 was a year of tragedy for Babbage; his father, his wife and two of his children all died that year. He own health gave way and he was advised to travel on the Continent. After his travels he returned near the end of 1828. Further attempts to obtain government support eventually resulted in the Duke of Wellington, the Chancellor of the Exchequer and other members of the government visiting Babbage and inspecting the work for themselves. By February 1830 the government had paid, or promised to pay, £9000 towards the project.

In 1830 Babbage published *Reflections on the Decline of Science in England*, a controversial work that resulted in the formation, one year later, of the British Association for the Advancement of Science. In 1834 Babbage published his most influential work *On the Economy of Machinery and Manufactures*, in which he proposed an early form of what today we call operational research.

The year 1834 was the one in which work stopped on the difference engine. By that time the government had put £17000 into the project and Babbage had put £6000 of his own money. For eight years from 1834 to 1842 the government would make no decision as to whether to continue support. In 1842 the decision not to proceed was taken by Robert Peel's government. Dubbey in [6] writes:-

*Babbage had every reason to feel aggrieved about his treatment by successive governments. They had failed to understand the immense possibilities of his work, ignored the advice of the most reputable scientists and engineers, procrastinated for*

*eight years before reaching a decision about the difference engine, misunderstood his motives and the sacrifices he had made, and ... failed to protect him from public slander and ridicule.*

By 1834 Babbage had completed the first drawings of the analytical engine, the forerunner of the modern electronic computer. His work on the difference engine had led him to a much more sophisticated idea. Although the analytic engine never progressed beyond detailed drawings, it is remarkably similar in logical components to a present day computer. Babbage describes five logical components, the store, the mill, the control, the input and the output. The store contains [4]:-

*... all the variables to be operated upon, as well as all those quantities which had arisen from the results of other operations.*

The mill is the analogue of the cpu in a modern computer and it is the place [4]:-

*... into which the quantities about to be operated upon are always bought.*

The control on the sequence of operations to be carried out was by a Jacquard loom type device. It was operated by punched cards and the punched cards contained the program for the particular task [4]:-

*Every set of cards made for any formula will at any future time recalculate the formula with whatever constants may be required.*

*Thus the Analytical Engine will possess a library of its own. Every set of cards once made will at any time reproduce the calculations for which it was first arranged.*

The store was to hold 1000 numbers each of 50 digits, but Babbage designed the analytic engine to effectively have infinite storage. This was done by outputting data to punched cards which could be read in again at a later stage when needed. Babbage decided, however, not to seek government support after his experiences with the difference engine.

Babbage visited Turin in 1840 and discussed his ideas with mathematicians there including Menabrea. During Babbage's visit, Menabrea collected all the material needed to describe the analytical engine and he published this in October 1842. Lady Ada Lovelace translated Menabrea's article into English and added notes considerably more extensive than the original memoir. This was published in 1843 and included [7]:-

*... elaborations on the points made by Menabrea, together with some complicated programs of her own, the most complex of these being one to calculate the sequence of Bernoulli numbers.*

Although Babbage never built an operational, mechanical computer, his design concepts have been proved correct and recently such a computer has been built

following Babbage's own design criteria.
See THIS LINK.

He wrote in 1851 (see [7]):-

*The drawings of the Analytical Engine have been made entirely at my own cost: I instituted a long series of experiments for the purpose of reducing the expense of its construction to limits which might be within the means I could myself afford to supply. I am now resigned to the necessity of abstaining from its construction...*

Despite this last statement, Babbage never did quite give up hope that the analytical engine would be built writing in 1864 in [4]:-

*... if I survive some few years longer, the Analytical Engine will exist...*

After Babbage's death a committee,whose members included Cayley and Clifford, was appointed by the British Association [12]:-

*… to report upon the feasibility of the design, recorded their opinion that its successful realisation might mark an epoch in the history of computation equally memorable with that of the introduction of logarithms…*

This was an underestimate. The construction of modern computers, logically similar to Babbage's design, have changed the whole of mathematics and it is even not an exaggeration to say that they have changed the whole world.

(😊)@@@@@@@@@@@@@@@@@@(😊)

# Computer Science

https://www.britannica.com/science/computer-science

A laptop computer Programmers may use computer science to design portable computers such as laptops.

**Computer Science**, the study of computers and computing, including their theoretical and algorithmic foundations, hardware and software, and their uses for processing information. The discipline of computer science includes the study of algorithms and data structures, computer and network design, modeling data and information processes, and artificial intelligence. Computer science draws some of its foundations from mathematics and engineering and therefore incorporates techniques from areas such as queueing theory, probability and statistics, and electronic circuit design. Computer science also makes heavy use of hypothesis testing and experimentation during the conceptualization, design, measurement, and refinement of new algorithms, information structures, and computer architectures.

**Explore the ProCon debate**

Computer science is considered as part of a family of five separate yet interrelated disciplines: computer engineering, computer science, information systems, information technology, and software engineering. This family has come to be known collectively as the discipline of computing. These five disciplines are interrelated in the sense that computing is their object of study, but they are separate since each has its own research perspective and curricular focus. (Since 1991 the Association for Computing Machinery [ACM], the IEEE Computer Society [IEEE-CS], and the Association for Information Systems [AIS] have collaborated to develop and update the taxonomy of these five interrelated disciplines and the

guidelines that educational institutions worldwide use for their undergraduate, graduate, and research programs.)

The major subfields of computer science include the traditional study of computer architecture, programming languages, and software development. However, they also include computational science (the use of algorithmic techniques for modeling scientific data), graphics and visualization, human-computer interaction, databases and information systems, networks, and the social and professional issues that are unique to the practice of computer science. As may be evident, some of these subfields overlap in their activities with other modern fields, such as bioinformatics and computational chemistry. These overlaps are the consequence of a tendency among computer scientists to recognize and act upon their field's many interdisciplinary connections.

## Development of computer science

Computer science emerged as an independent discipline in the early 1960s, although the electronic digital computer that is the object of its study was invented some two decades earlier. The roots of computer science lie primarily in the related fields of mathematics, electrical engineering, physics, and management information systems.

Mathematics is the source of two key concepts in the development of the computer—the idea that all information can be represented as sequences of zeros and ones and the abstract notion of a "stored program." In the binary number system, numbers are represented by a sequence of the binary digits 0 and 1 in the same way that numbers in the familiar decimal system are represented using the digits 0 through 9. The relative ease with which two states (e.g., high and low voltage) can be realized in electrical and electronic devices led naturally to the binary digit, or bit, becoming the basic unit of data storage and transmission in a computer system.

Electrical engineering provides the basics of circuit design—namely, the idea that electrical impulses input to a circuit can be combined using Boolean algebra to produce arbitrary outputs. (The Boolean algebra developed in the 19th century supplied a formalism for designing a circuit with binary input values of zeros and ones [false or true, respectively, in the terminology of logic] to yield any desired combination of zeros and ones as output.) The invention of the transistor and the miniaturization of circuits, along with the invention of electronic, magnetic, and optical media for the storage and transmission of information, resulted from advances in electrical engineering and physics.

Management information systems, originally called data processing systems, provided early ideas from which various computer science concepts such as sorting, searching, databases, information retrieval, and graphical user interfaces evolved. Large corporations housed computers that stored information that was central to the activities of running a business—payroll, accounting, inventory management, production control, shipping, and receiving.

**British mathematician Alan Turing conceptualized the Turing machine.**

Theoretical work on computability, which began in the 1930s, provided the needed extension of these advances to the design of whole machines; a milestone was the 1936 specification of the Turing machine (a theoretical computational model that carries out instructions represented as a series of zeros and ones) by the British mathematician Alan Turing and his proof of the model's computational power. Another breakthrough was the concept of the stored-program computer, usually credited to Hungarian American mathematician John von Neumann. These are the origins of the computer science field that later became known as architecture and organization.

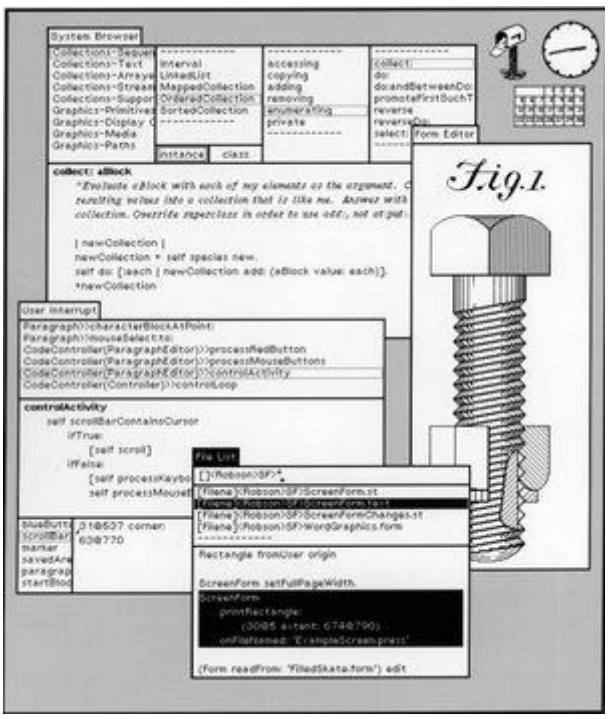
In the 1950s, most computer users worked either in scientific research labs or in large corporations. The former group used computers to help them make complex mathematical calculations (e.g., missile trajectories), while the latter group used computers to manage large amounts of corporate data (e.g., payrolls and inventories). Both groups quickly learned that writing programs in the machine language of zeros and ones was not practical or reliable. This discovery led to the development of assembly language in the early 1950s, which allows programmers to use symbols for instructions (e.g., ADD for addition) and variables (e.g., *X*). Another program, known as an assembler, translated these symbolic programs into an equivalent binary program whose steps the computer could carry out, or "execute."

Other system software elements known as linking loaders were developed to combine pieces of assembled code and load them into the computer's memory, where they could be executed. The concept of linking separate pieces of code was important, since it allowed "libraries" of programs for carrying out common tasks to be reused. This was a first step in the development of the computer science field called software engineering.

Later in the 1950s, assembly language was found to be so cumbersome that the development of high-level languages (closer to natural languages) began to support easier, faster programming. FORTRAN emerged as the main high-level language for scientific programming, while COBOL became the main language for business programming. These languages carried with them the need for different software, called compilers, that translate high-level language programs into machine code. As programming languages became more powerful and abstract, building compilers that create high-quality machine code and that are efficient in terms of execution speed and storage consumption became a challenging computer science problem. The design and implementation of high-level languages is at the heart of the computer science field called programming languages.

Increasing use of computers in the early 1960s provided the impetus for the development of the first operating systems, which consisted of system-resident software that automatically handled input and output and the execution of programs called "jobs." The demand for better computational techniques led to a resurgence of interest in numerical methods and their analysis, an activity that expanded so widely that it became known as computational science.

The 1970s and '80s saw the emergence of powerful computer graphics devices, both for scientific modeling and other visual activities. (Computerized graphical devices were introduced in the early 1950s with the display of crude images on paper plots and cathode-ray tube [CRT] screens.) Expensive hardware and the limited availability of software kept the field from growing until the early 1980s, when the computer memory required for bitmap graphics (in which an image is made up of small rectangular pixels) became more affordable. Bitmap technology, together with high-resolution display screens and the development of graphics standards that make software less machine-dependent, has led to the explosive growth of the field. Support for all these activities evolved into the field of computer science known as graphics and visual computing.




**The Xerox Alto was the first computer to use graphical icons and a mouse to control the system—the first graphical user interface (GUI).**

Closely related to this field is the design and analysis of systems that interact directly with users who are carrying out various computational tasks. These systems came into wide use during the 1980s and ’90s, when line-edited interactions with users were replaced by graphical user interfaces (GUIs). GUI design, which was pioneered by Xerox and was later picked up by Apple (Macintosh) and finally by Microsoft (Windows), is important because it constitutes what people see and do when they interact with a computing device. The design of appropriate user interfaces for all types of users has evolved into the computer science field known as human-computer interaction (HCI).

The field of computer architecture and organization has also evolved dramatically since the first stored-program computers were developed in the 1950s. So called time-sharing systems emerged in the 1960s to allow several users to run programs at the same time from different terminals that were hard-wired to the computer. The 1970s saw the development of the first wide-area computer networks (WANs) and protocols for transferring information at high speeds between computers separated by large distances. As these activities evolved, they coalesced into the computer science field called networking and communications. A major accomplishment of this field was the development of the Internet.

The idea that instructions, as well as data, could be stored in a computer’s memory was critical to fundamental discoveries about the theoretical behavior of algorithms. That is, questions such as, “What can/cannot be computed?” have been formally addressed using these abstract ideas. These discoveries were the origin of the computer science field known as algorithms and complexity. A key part of this field is the study and application of data structures that are appropriate to different applications. Data structures, along with the development of optimal algorithms for inserting, deleting, and locating data in such structures, are a major concern of computer scientists because they are so heavily used in computer software, most notably in compilers, operating systems, file systems, and search engines.

In the 1960s the invention of magnetic disk storage provided rapid access to data located at an arbitrary place on the disk. This invention led not only to more cleverly designed file systems but also to the development of database and information retrieval systems, which later became essential for storing, retrieving, and transmitting large amounts and wide varieties of data across the Internet. This field of computer science is known as information management.

Another long-term goal of computer science research is the creation of computing machines and robotic devices that can carry out tasks that are typically thought of as requiring human intelligence. Such tasks include moving, seeing, hearing, speaking, understanding natural language, thinking, and even exhibiting human emotions. The computer science field of intelligent systems, originally known as artificial intelligence (AI), actually predates the first electronic computers in the 1940s, although the term *artificial intelligence* was not coined until 1956.

Three developments in computing in the early part of the 21st century—mobile computing, client-server computing, and computer hacking—contributed to the emergence of three new fields in computer science: platform-based development, parallel and distributed computing, and security and information assurance. Platform-based development is the study of the special needs of mobile devices, their operating systems, and their applications. Parallel and distributed computing concerns the development of architectures and programming languages that support the development of algorithms whose components can run simultaneously and asynchronously (rather than sequentially), in order to make better use of time

and space. Security and information assurance deals with the design of computing systems and software that protects the integrity and security of data, as well as the privacy of individuals who are characterized by that data.

Finally, a particular concern of computer science throughout its history is the unique societal impact that accompanies computer science research and technological advancements. With the emergence of the Internet in the 1980s, for example, software developers needed to address important issues related to information security, personal privacy, and system reliability. In addition, the question of whether computer software constitutes intellectual property and the related question "Who owns it?" gave rise to a whole new legal area of licensing and licensing standards that applied to software and related artifacts. These concerns and others form the basis of social and professional issues of computer science, and they appear in almost all the other fields identified above.

So, to summarize, the discipline of computer science has evolved into the following 15 distinct fields:

- Algorithms and complexity
- Architecture and organization
- Computational science
- Graphics and visual computing
- Human-computer interaction
- Information management
- Intelligent systems
- Networking and communication
- Operating systems
- Parallel and distributed computing
- Platform-based development
- Programming languages
- Security and information assurance
- Software engineering
- Social and professional issues

Computer science continues to have strong mathematical and engineering roots. Computer science bachelor's, master's, and doctoral degree programs are routinely offered by postsecondary academic institutions, and these programs require students to complete appropriate mathematics and engineering courses, depending on their area of focus. For example, all undergraduate computer science majors must study discrete mathematics (logic, combinatorics, and elementary graph theory). Many programs also require students to complete courses in calculus, statistics, numerical analysis, physics, and principles of engineering early in their studies.

## Algorithms and complexity

An algorithm is a specific procedure for solving a well-defined computational problem. The development and analysis of algorithms is fundamental to all aspects of computer science: artificial intelligence, databases, graphics, networking, operating systems, security, and so on. Algorithm development is more than just programming. It requires an understanding of the alternatives available for solving a computational problem,

including the hardware, networking, programming language, and performance constraints that accompany any particular solution. It also requires understanding what it means for an algorithm to be "correct" in the sense that it fully and efficiently solves the problem at hand.

An accompanying notion is the design of a particular data structure that enables an algorithm to run efficiently. The importance of data structures stems from the fact that the main memory of a computer (where the data is stored) is linear, consisting of a sequence of memory cells that are serially numbered 0, 1, 2,…. Thus, the simplest data structure is a linear array, in which adjacent elements are numbered with consecutive integer "indexes" and an element's value is accessed by its unique index. An array can be used, for example, to store a list of names, and efficient methods are needed to efficiently search for and retrieve a particular name from the array. For example, sorting the list into alphabetical order permits a so-called binary search technique to be used, in which the remainder of the list to be searched at each step is cut in half. This search technique is similar to searching a telephone book for a particular name. Knowing that the book is in alphabetical order allows one to turn quickly to a page that is close to the page containing the desired name. Many algorithms have been developed for sorting and searching lists of data efficiently.

Although data items are stored consecutively in memory, they may be linked together by pointers (essentially, memory addresses stored with an item to indicate where the next item or items in the structure are found) so that the data can be organized in ways similar to those in which they will be accessed. The simplest such structure is called the linked list, in which noncontiguously stored items may be accessed in a pre-specified order by following the pointers from one item in the list to the next. The list may be circular, with the last item pointing to the first, or each element may have pointers in both directions to form a doubly linked list. Algorithms have been developed for efficiently manipulating such lists by searching for, inserting, and removing items.

Pointers also provide the ability to implement more complex data structures. A graph, for example, is a set of nodes (items) and links (known as edges) that connect pairs of items. Such a graph might represent a set of cities and the highways joining them, the layout of circuit elements and connecting wires on a memory chip, or the configuration of persons interacting via a social network. Typical graph algorithms include graph traversal strategies, such as how to follow the links from node to node (perhaps searching for a node with a particular property) in a way that each node is visited only once. A related problem is the determination of the shortest path between two given nodes on an arbitrary graph. (*See* graph theory.) A problem of practical interest in network algorithms, for instance, is to determine how many "broken" links can be tolerated before communications begin to fail. Similarly, in very-large-scale integration (VLSI) chip design it is important to know whether the graph representing a circuit is planar, that is, whether it can be drawn in two dimensions without any links crossing (wires touching).

The (computational) complexity of an algorithm is a measure of the amount of computing resources (time and space) that a particular algorithm consumes when it runs. Computer scientists use mathematical measures of complexity that allow them to predict, before writing the code, how fast an algorithm will run and how

much memory it will require. Such predictions are important guides for programmers implementing and selecting algorithms for real-world applications.

Computational complexity is a continuum, in that some algorithms require linear time (that is, the time required increases directly with the number of items or nodes in the list, graph, or network being processed), whereas others require quadratic or even exponential time to complete (that is, the time required increases with the number of items squared or with the exponential of that number). At the far end of this continuum lie the murky seas of intractable problems—those whose solutions cannot be efficiently implemented. For these problems, computer scientists seek to find heuristic algorithms that can almost solve the problem and run in a reasonable amount of time.

Further away still are those algorithmic problems that can be stated but are not solvable; that is, one can prove that no program can be written to solve the problem. A classic example of an unsolvable algorithmic problem is the halting problem, which states that no program can be written that can predict whether or not any other program halts after a finite number of steps. The unsolvability of the halting problem has immediate practical bearing on software development. For instance, it would be frivolous to try to develop a software tool that predicts whether another program being developed has an infinite loop in it (although having such a tool would be immensely beneficial).

## Architecture and organization

Computer architecture deals with the design of computers, data storage devices, and networking components that store and run programs, transmit data, and drive interactions between computers, across networks, and with users. Computer architects use parallelism and various strategies for memory organization to design computing systems with very high performance. Computer architecture requires strong communication between computer scientists and computer engineers, since they both focus fundamentally on hardware design.

At its most fundamental level, a computer consists of a control unit, an arithmetic logic unit (ALU), a memory unit, and input/output (I/O) controllers. The ALU performs simple addition, subtraction, multiplication, division, and logic operations, such as OR and AND. The memory stores the program's instructions and data. The control unit fetches data and instructions from memory and uses operations of the ALU to carry out those instructions using that data. (The control unit and ALU together are referred to as the central processing unit [CPU].) When an input or output instruction is encountered, the control unit transfers the data between the memory and the designated I/O controller. The operational speed of the CPU primarily determines the speed of the computer as a whole. All of these components—the control unit, the ALU, the memory, and the I/O controllers—are realized with transistor circuits.

Computers also have another level of memory called a cache, a small, extremely fast (compared with the main memory, or random access memory [RAM]) unit that can be used to store information that is urgently or frequently needed. Current research includes cache design and algorithms that can predict what data is likely to be needed next and preload it into the cache for improved performance.

I/O controllers connect the computer to specific input devices (such as keyboards and touch screen displays) for feeding information to the memory, and output devices (such as printers and displays) for transmitting information from the memory to users. Additional I/O controllers connect the computer to a network via ports that provide the conduit through which data flows when the computer is connected to the Internet.

**USB flash drive inserted in a laptop Devices such as flash drives played an important role in backing up and transferring data.**

Linked to the I/O controllers are secondary storage devices, such as a disk drive, that are slower and have a larger capacity than main or cache memory. Disk drives are used for maintaining permanent data. They can be either permanently or temporarily attached to the computer in the form of a compact disc (CD), a digital video disc (DVD), or a memory stick (also called a flash drive).

The operation of a computer, once a program and some data have been loaded into RAM, takes place as follows. The first instruction is transferred from RAM into the control unit and interpreted by the hardware circuitry. For instance, suppose that the instruction is a string of bits that is the code for LOAD 10. This instruction loads the contents of memory location 10 into the ALU. The next instruction, say ADD 15, is fetched. The control unit then loads the contents of memory location 15 into the ALU and adds it to the number already there. Finally, the instruction STORE 20 would store that sum into location 20. At this level, the operation of a computer is not much different from that of a pocket calculator.

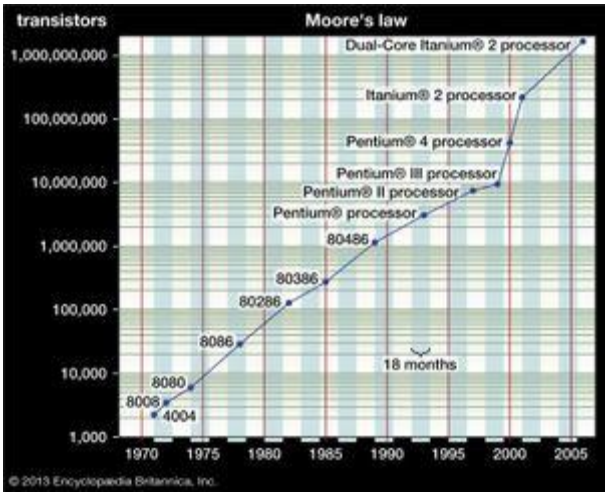
In general, programs are not just lengthy sequences of LOAD, STORE, and arithmetic operations. Most importantly, computer languages include conditional instructions—essentially, rules that say, "If memory location *n* satisfies condition *a*, do instruction number *x* next, otherwise do instruction *y*." This allows the course of a program to be determined by the results of previous operations—a critically important ability.

Finally, programs typically contain sequences of instructions that are repeated a number of times until a predetermined condition becomes true. Such a sequence is called a loop. For example, a loop would be needed to compute the sum of the first *n* integers, where *n* is a value stored in a separate memory location. Computer architectures that can execute sequences of instructions, conditional instructions, and loops are called "Turing complete," which means that they can carry out the execution of any algorithm that can be defined. Turing completeness is a fundamental and essential characteristic of any computer organization.

Logic design is the area of computer science that deals with the design of electronic circuits using the fundamental principles and properties of logic (*see* Boolean algebra) to carry out the operations of the control unit, the ALU, the I/O controllers, and other hardware. Each logical function (AND, OR, and NOT) is realized by a particular type of device called a gate. For example, the addition circuit of the ALU has inputs corresponding to all the bits of the two numbers to be added and outputs corresponding to the bits of the sum. The arrangement of wires and gates that link inputs to outputs is determined by the mathematical definition of addition. The design of the control unit provides the circuits that interpret instructions. Due to the need for efficiency, logic design must also optimize the circuitry to function with maximum speed and has a minimum number of gates and circuits.

An important area related to architecture is the design of microprocessors, which are complete CPUs—control unit, ALU, and memory—on a single integrated circuit chip. Additional memory and I/O control circuitry are linked to this chip to form a complete computer. These thumbnail-sized devices contain millions of transistors that implement the processing and memory units of modern computers.

VLSI microprocessor design occurs in a number of stages, which include creating the initial functional or behavioral specification, encoding this specification into a hardware description language, and breaking down the design into modules and generating sizes and shapes for the eventual chip components. It also involves chip planning, which includes building a "floor plan" to indicate where on the chip each component should be placed and connected to other components. Computer scientists are also involved in creating the computer-aided design (CAD) tools that support engineers in the various stages of chip design and in developing the necessary theoretical results, such as how to efficiently design a floor plan with near-minimal area that satisfies the given constraints.




**Moore's law Gordon Moore observed that the number of transistors on a computer chip was doubling about every 18–24 months. As shown in the logarithmic graph of the number of transistors on Intel's processors at the time of their introduction, his "law" was being obeyed.**

Advances in integrated circuit technology have been incredible. For example, in 1971 the first microprocessor chip (Intel Corporation's 4004) had only 2,300 transistors, in 1993 Intel's Pentium chip had more than 3 million transistors, and by 2000 the number of transistors on such a chip was about 50 million. The Power7 chip introduced in 2010 by IBM contained approximately 1 billion transistors. The

phenomenon of the number of transistors in an integrated circuit doubling about every two years is widely known as Moore's law.

Fault tolerance is the ability of a computer to continue operation when one or more of its components fails. To ensure fault tolerance, key components are often replicated so that the backup component can take over if needed. Such applications as aircraft control and manufacturing process control run on systems with backup processors ready to take over if the main processor fails, and the backup systems often run in parallel so the transition is smooth. If the systems are critical in that their failure would be potentially disastrous (as in aircraft control), incompatible outcomes collected from replicated processes running in parallel on separate machines are resolved by a voting mechanism. Computer scientists are involved in the analysis of such replicated systems, providing theoretical approaches to estimating the reliability achieved by a given configuration and processor parameters, such as average time between failures and average time required to repair the processor. Fault tolerance is also a desirable feature in distributed systems and networks. For example, an advantage of a distributed database is that data replicated on different network hosts can provide a natural backup mechanism when one host fails.

## Computational science

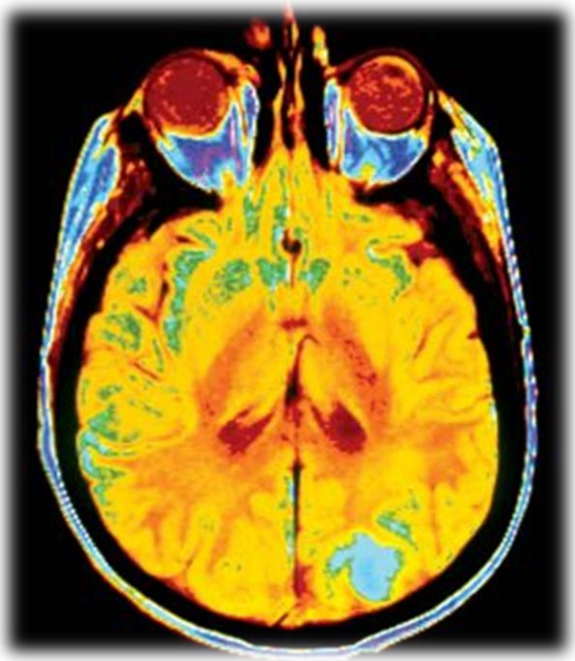
Computational science applies computer simulation, scientific visualization, mathematical modeling, algorithms, data structures, networking, database design, symbolic computation, and high-performance computing to help advance the goals of various disciplines. These disciplines include biology, chemistry, fluid dynamics, archaeology, finance, sociology, and forensics. Computational science has evolved rapidly, especially because of the dramatic growth in the volume of data transmitted from scientific instruments. This phenomenon has been called the "big data" problem.

The mathematical methods needed for computational science require the transformation of equations and functions from the continuous to the discrete. For example, the computer integration of a function over an interval is accomplished not by applying integral calculus but rather by approximating the area under the function graph as a sum of the areas obtained from evaluating the function at discrete points. Similarly, the solution of a differential equation is obtained as a sequence of discrete points determined by approximating the true solution curve by a sequence of tangential line segments. When discretized in this way, many problems can be recast as an equation involving a matrix (a rectangular array of numbers) solvable using linear algebra. Numerical analysis is the study of such computational methods. Several factors must be considered when applying numerical methods: (1) the conditions under which the method yields a solution, (2) the accuracy of the solution, (3) whether the solution process is stable (i.e., does not exhibit error growth), and (4) the computational complexity (in the sense described above) of obtaining a solution of the desired accuracy.

The requirements of big-data scientific problems, including the solution of ever larger systems of equations, engage the use of large and powerful arrays of processors (called multiprocessors or supercomputers) that allow many calculations to proceed in parallel by assigning them to separate processing elements. These activities have sparked much interest in parallel computer architecture and algorithms that can be carried out efficiently on such machines.

## Graphics and visual computing

Graphics and visual computing is the field that deals with the display and control of images on a computer screen. This field encompasses the efficient implementation of four interrelated computational tasks: rendering, modeling, animation, and visualization. Graphics techniques incorporate principles of linear algebra, numerical integration, computational geometry, special-purpose hardware, file formats, and graphical user interfaces (GUIs) to accomplish these complex tasks.



**Brain cancer; magnetic resonance imaging (MRI)An image produced by magnetic resonance imaging (MRI) of a human brain affected by cancer. The bright blue area indicates that the cancer spread to the occipital lobe (lower right).**

Applications of graphics include CAD, fine arts, medical imaging, scientific data visualization, and video games. CAD systems allow the computer to be used for designing objects ranging from automobile parts to bridges to computer chips by providing an interactive drawing tool and an engineering interface to simulation and analysis tools. Fine arts applications allow artists to use the computer screen as a medium to create images, cinematographic special effects, animated cartoons, and television commercials. Medical imaging applications involve the visualization of data obtained from technologies such as X-rays and magnetic resonance imaging (MRIs) to assist doctors in diagnosing medical conditions. Scientific visualization uses massive amounts of data to define simulations of scientific phenomena, such as ocean modeling, to produce pictures that provide more insight into the phenomena than would tables of numbers. Graphics also provide realistic visualizations for video gaming, flight simulation, and other representations of reality or fantasy. The term *virtual reality* has been coined to refer to any interaction with a computer-simulated virtual world.

A challenge for computer graphics is the development of efficient algorithms that manipulate the myriad of lines, triangles, and polygons that make up a computer image. In order for realistic on-screen images to be presented, each object must be rendered as a set of planar units. Edges must be smoothed and textured so that their underlying construction from polygons is not obvious to the naked eye. In many applications, still pictures are inadequate, and rapid display of real-time

images is required. Both extremely efficient algorithms and state-of-the-art hardware are needed to accomplish real-time animation. (For more technical details of graphics displays, *see* computer graphics.)

## Human-computer interaction

Human-computer interaction (HCI) is concerned with designing effective interaction between users and computers and the construction of interfaces that support this interaction. HCI occurs at an interface that includes both software and hardware. User interface design impacts the life cycle of software, so it should occur early in the design process. Because user interfaces must accommodate a variety of user styles and capabilities, HCI research draws on several disciplines including psychology, sociology, anthropology, and engineering. In the 1960s, user interfaces consisted of computer consoles that allowed an operator directly to type commands that could be executed immediately or at some future time. With the advent of more user-friendly personal computers in the 1980s, user interfaces became more sophisticated, so that the user could "point and click" to send a command to the operating system.

Thus, the field of HCI emerged to model, develop, and measure the effectiveness of various types of interfaces between a computer application and the person accessing its services. GUIs enable users to communicate with the computer by such simple means as pointing to an icon with a mouse or touching it with a stylus or forefinger. This technology also supports windowing environments on a computer screen, which allow users to work with different applications simultaneously, one in each window.

## Information management

Information management (IM) is primarily concerned with the capture, digitization, representation, organization, transformation, and presentation of information. Because a computer's main memory provides only temporary storage, computers are equipped with auxiliary disk storage devices that permanently store data. These devices are characterized by having much higher capacity than main memory but slower read/write (access) speed. Data stored on a disk must be read into main memory before it can be processed. A major goal of IM systems, therefore, is to develop efficient algorithms to store and retrieve specific data for processing.

IM systems comprise databases and algorithms for the efficient storage, retrieval, updating, and deleting of specific items in the database. The underlying structure of a database is a set of files residing permanently on a disk storage device. Each file can be further broken down into a series of records, which contains individual data items, or fields. Each field gives the value of some property (or attribute) of the entity represented by a record. For example, a personnel file may contain a series of records, one for each individual in the organization, and each record would contain fields that contain that person's name, address, phone number, email address, and so forth.

Many file systems are sequential, meaning that successive records are processed in the order in which they are stored, starting from the beginning and proceeding to the end. This file structure was particularly popular in the early days of computing, when files were stored on reels of magnetic tape and these reels could be processed

only in a sequential manner. Sequential files are generally stored in some sorted order (e.g., alphabetic) for printing of reports (e.g., a telephone directory) and for efficient processing of batches of transactions. Banking transactions (deposits and withdrawals), for instance, might be sorted in the same order as the accounts file, so that as each transaction is read the system need only scan ahead to find the accounts record to which it applies.

With modern storage systems, it is possible to access any data record in a random fashion. To facilitate efficient random access, the data records in a file are stored with indexes called keys. An index of a file is much like an index of a book; it contains a key for each record in the file along with the location where the record is stored. Since indexes might be long, they are usually structured in some hierarchical fashion so that they can be navigated efficiently. The top level of an index, for example, might contain locations of (point to) indexes to items beginning with the letters A, B, etc. The A index itself may contain not locations of data items but pointers to indexes of items beginning with the letters Ab, Ac, and so on. Locating the index for the desired record by traversing a treelike structure is quite efficient.

Many applications require access to many independent files containing related and even overlapping data. Their information management activities frequently require data from several files to be linked, and hence the need for a database model emerges. Historically, three different types of database models have been developed to support the linkage of records of different types: (1) the hierarchical model, in which record types are linked in a treelike structure (e.g., employee records might be grouped under records describing the departments in which employees work), (2) the network model, in which arbitrary linkages of record types may be created (e.g., employee records might be linked on one hand to employees' departments and on the other hand to their supervisors—that is, other employees), and (3) the relational model, in which all data are represented in simple tabular form.

In the relational model, each individual entry is described by the set of its attribute values (called a relation), stored in one row of the table. This linkage of *n* attribute values to provide a meaningful description of a real-world entity or a relationship among such entities forms a mathematical *n*-tuple. The relational model also supports queries (requests for information) that involve several tables by providing automatic linkage across tables by means of a "join" operation that combines records with identical values of common attributes. Payroll data, for example, can be stored in one table and personnel benefits data in another; complete information on an employee could be obtained by joining the two tables using the employee's unique identification number as a common attribute.

To support database processing, a software artifact known as a database management system (DBMS) is required to manage the data and provide the user with commands to retrieve information from the database. For example, a widely used DBMS that supports the relational model is MySQL.

Another development in database technology is to incorporate the object concept. In object-oriented databases, all data are objects. Objects may be linked together by an "is-part-of" relationship to represent larger, composite objects. Data describing a truck, for instance, may be stored as a composite of a particular engine, chassis, drive train, and so forth. Classes of objects may form a hierarchy in which

individual objects may inherit properties from objects farther up in the hierarchy. For example, objects of the class "motorized vehicle" all have an engine; members of the subclasses "truck" or "airplane" will then also have an engine.

NoSQL, or non-relational databases, have also emerged. These databases are different from the classic relational databases because they do not require fixed tables. Many of them are document-oriented databases, in which voice, music, images, and video clips are stored along with traditional textual information. An important subset of NoSQL are the XML databases, which are widely used in the development of Android smartphone and tablet applications.

Data integrity refers to designing a DBMS that ensures the correctness and stability of its data across all applications that access the system. When a database is designed, integrity checking is enabled by specifying the data type of each column in the table. For example, if an identification number is specified to be nine digits, the DBMS will reject an update attempting to assign a value with more or fewer digits or one including an alphabetic character. Another type of integrity, known as referential integrity, requires that each entity referenced by some other entity must itself exist in the database. For example, if an airline reservation is requested for a particular flight number, then the flight referenced by that number must actually exist.

Access to a database by multiple simultaneous users requires that the DBMS include a concurrency control mechanism (called locking) to maintain integrity whenever two different users attempt to access the same data at the same time. For example, two travel agents may try to book the last seat on a plane at more or less the same time. Without concurrency control, both may think they have succeeded, though only one booking is actually entered into the database.

A key concept in studying concurrency control and the maintenance of data integrity is the transaction, defined as an indivisible operation that transforms the database from one state into another. To illustrate, consider an electronic transfer of funds of $5 from bank account A to account B. The operation that deducts $5 from account A leaves the database without integrity since the total over all accounts is $5 short. Similarly, the operation that adds $5 to account B in itself makes the total $5 too much. Combining these two operations into a single transaction, however, maintains data integrity. The key here is to ensure that only complete transactions are applied to the data and that multiple concurrent transactions are executed using locking so that serializing them would produce the same result. A transaction-oriented control mechanism for database access becomes difficult in the case of a long transaction, for example, when several engineers are working, perhaps over the course of several days, on a product design that may not exhibit data integrity until the project is complete.

As mentioned previously, a database may be distributed in that its data can be spread among different host computers on a network. If the distributed data contains duplicates, the concurrency control problem is more complex. Distributed databases must have a distributed DBMS to provide overall control of queries and updates in a manner that does not require that the user know the location of the data. A closely related concept is interoperability, meaning the ability of the user of one member of a group of disparate systems (all having the same functionality) to work with any of the systems of the group with equal ease and via the same interface.

## Intelligent systems

**ASIMOThe two-legged humanoid robot developed by the Honda Motor Company in the early 21st century could walk independently and climb or descend stairs.**

Artificial intelligence (AI) is an area of research that goes back to the very beginnings of computer science. The idea of building a machine that can perform tasks perceived as requiring human intelligence is an attractive one. The tasks that have been studied from this point of view include game playing, language translation, natural language understanding, fault diagnosis, robotics, and supplying expert advice.

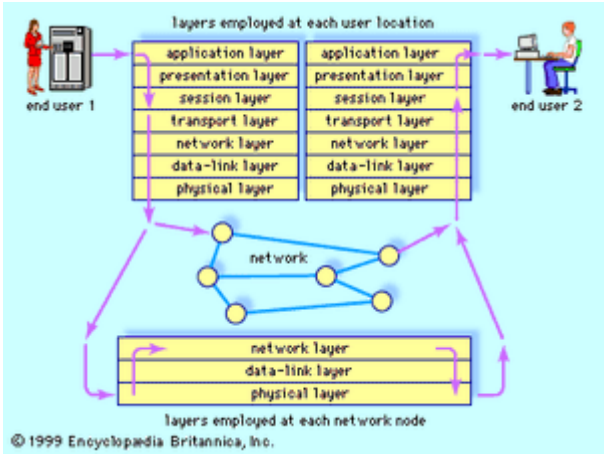
Since the late 20th century, the field of intelligent systems has focused on the support of everyday applications—email, word processing, and search—using nontraditional techniques. These techniques include the design and analysis of autonomous agents that perceive their environment and interact rationally with it. The solutions rely on a broad set of knowledge-representation schemes, problem-solving mechanisms, and learning strategies. They deal with sensing (e.g., speech recognition, natural language understanding, and computer vision), problem-solving (e.g., search and planning), acting (e.g., robotics), and the architectures needed to support them (e.g,. agents and multi-agents).

## Networking and communication

The field of networking and communication includes the analysis, design, implementation, and use of local, wide-area, and mobile networks that link computers together. The Internet itself is a network that makes it feasible for nearly all computers in the world to communicate.

A computer network links computers together via a combination of infrared light signals, radio wave transmissions, telephone lines, television cables, and satellite links. The challenge for computer scientists has been to

develop protocols (standardized rules for the format and exchange of messages) that allow processes running on host computers to interpret the signals they receive and to engage in meaningful "conversations" in order to accomplish tasks on behalf of users. Network protocols also include flow control, which keeps a data sender from swamping a receiver with messages that it has no time to process or space to store, and error control, which involves transmission error detection and automatic resending of messages to correct such errors. (For some of the technical details of error detection and correction, *see* information theory.)




**Open systems interconnection (OSI)Established in 1983 by the International Organization for Standardization, the OSI model divides network protocols (standardized procedures for exchanging information) into seven functional "layers." This communications architecture enables end users employing different operating systems or working in different networks to communicate quickly and correctly.**

The standardization of protocols is an international effort. Since it would otherwise be impossible for different kinds of machines and operating systems to communicate with one another, the key concern has been that system components (computers) be "open." This terminology comes from the open systems interconnection (OSI) communication standards, established by the International Organization for Standardization. The OSI reference model specifies network protocol standards in seven layers. Each layer is defined by the functions it relies upon from the layer below it and by the services it provides to the layer above it.

At the bottom of the protocol lies the physical layer, containing rules for the transport of bits across a physical link. The data-link layer handles standard-sized "packets" of data and adds reliability in the form of error detection and flow control bits. The network and transport layers break messages into the standard-size packets and route them to their destinations. The session layer supports interactions between applications on two communicating machines. For example, it provides a mechanism with which to insert checkpoints (saving the current status of a task) into a long file transfer so that, in case of a failure, only the data after the last checkpoint need to be retransmitted. The presentation layer is concerned with functions that encode data, so that heterogeneous systems may engage in meaningful communication. At the highest level are protocols that support specific applications. An example of such an application is the file transfer protocol (FTP), which governs the transfer of files from one host to another.

The development of networks and communication protocols has also spawned distributed systems, in which computers linked in a network share data and processing tasks. A distributed database system, for example, has a database spread among (or replicated at) different network sites. Data are replicated at "mirror sites," and replication can improve availability and reliability. A distributed DBMS manages a database whose components are distributed across several computers on a network.

A client-server network is a distributed system in which the database resides on one computer (the server) and the users connect to this computer over the network from their own computers (the clients). The server provides data and responds to requests from each client, while each client accesses the data on the server in a way that is independent and ignorant of the presence of other clients accessing the same database. Client-server systems require that individual actions from several clients to the same part of the server's database be synchronized, so that conflicts are resolved in a reasonable way. For example, airline reservations are implemented using a client-server model. The server contains all the data about upcoming flights, such as current bookings and seat assignments. Each client wants to access this data for the purpose of booking a flight, obtaining a seat assignment, and paying for the flight. During this process, it is likely that two or more client requests want to access the same flight and that there is only one seat left to be assigned. The software must synchronize these two requests so that the remaining seat is assigned in a rational way (usually to the person who made the request first).

Another popular type of distributed system is the peer-to-peer network. Unlike client-server networks, a peer-to-peer network assumes that each computer (user) connected to it can act both as a client and as a server; thus, everyone on the network is a peer. This strategy makes sense for groups that share audio collections on the Internet and for organizing social networks such as LinkedIn and Facebook. Each person connected to such a network both receives information from others and shares his or her own information with others.

## Operating systems

An operating system is a specialized collection of software that stands between a computer's hardware architecture and its applications. It performs a number of fundamental activities such as file system management, process scheduling, memory allocation, network interfacing, and resource sharing among the computer's users. Operating systems have evolved in their complexity over time, beginning with the earliest computers in the 1960s.

With early computers, the user typed programs onto punched tape or cards, which were read into the computer, assembled or compiled, and run. The results were then transmitted to a printer or a magnetic tape. These early operating systems engaged in batch processing; i.e., handling sequences of jobs that are compiled and executed one at a time without intervention by the user. Accompanying each job in a batch were instructions to the operating system (OS) detailing the resources needed by the job, such as the amount of CPU time required, the files needed, and the storage devices on which the files resided. From these beginnings came the key concept of an operating system as a resource allocator. This role became more important with the rise of multiprogramming, in which several jobs reside in the

computer simultaneously and share resources, for example, by being allocated fixed amounts of CPU time in turn. More sophisticated hardware allowed one job to be reading data while another wrote to a printer and still another performed computations. The operating system thus managed these tasks in such a way that all the jobs were completed without interfering with one another.

The advent of time sharing, in which users enter commands and receive results directly at a terminal, added more tasks to the operating system. Processes known as terminal handlers were needed, along with mechanisms such as interrupts (to get the attention of the operating system to handle urgent tasks) and buffers (for temporary storage of data during input/output to make the transfer run more smoothly). Modern large computers interact with hundreds of users simultaneously, giving each one the perception of being the sole user.

Another area of operating system research is the design of virtual memory. Virtual memory is a scheme that gives users the illusion of working with a large block of contiguous memory space (perhaps even larger than real memory), when in actuality most of their work is on auxiliary storage (disk). Fixed-size blocks (pages) or variable-size blocks (segments) of the job are read into main memory as needed. Questions such as how much main memory space to allocate to users and which pages or segments should be returned to disk ("swapped out") to make room for incoming pages or segments must be addressed in order for the system to execute jobs efficiently.

The first commercially viable operating systems were developed by IBM in the 1960s and were called OS/360 and DOS/360. Unix was developed at Bell Laboratories in the early 1970s and since has spawned many variants, including Linux, Berkeley Unix, GNU, and Apple's iOS. Operating systems developed for the first personal computers in the 1980s included IBM's (and later Microsoft's) DOS, which evolved into various flavors of Windows. An important 21st-century development in operating systems was that they became increasingly machine-independent.

## Parallel and distributed computing

The simultaneous growth in availability of big data and in the number of simultaneous users on the Internet places particular pressure on the need to carry out computing tasks "in parallel," or simultaneously. Parallel and distributed computing occurs across many different topic areas in computer science, including algorithms, computer architecture, networks, operating systems, and software engineering. During the early 21st century there was explosive growth in multiprocessor design and other strategies for complex applications to run faster. Parallel and distributed computing builds on fundamental systems concepts, such as concurrency, mutual exclusion, consistency in state/memory manipulation, message-passing, and shared-memory models.

Creating a multiprocessor from a number of single CPUs requires physical links and a mechanism for communication among the processors so that they may operate in parallel. Tightly coupled multiprocessors share memory and hence may communicate by storing information in memory accessible by all processors. Loosely coupled multiprocessors, including computer networks, communicate by sending messages to each other across the physical links. Computer scientists have investigated various multiprocessor architectures. For example, the possible

configurations in which hundreds or even thousands of processors may be linked together are examined to find the geometry that supports the most efficient system throughput. A much-studied topology is the hypercube, in which each processor is connected directly to some fixed number of neighbors: two for the two-dimensional square, three for the three-dimensional cube, and similarly for the higher-dimensional hypercubes. Computer scientists also investigate methods for carrying out computations on such multiprocessor machines (e.g., algorithms to make optimal use of the architecture and techniques to avoid conflicts in data transmission). The machine-resident software that makes possible the use of a particular machine, in particular its operating system, is an integral part of this investigation.

Concurrency refers to the execution of more than one procedure at the same time (perhaps with the access of shared data), either truly simultaneously (as on a multiprocessor) or in an unpredictably interleaved order. Modern programming languages such as Java include both encapsulation and features called "threads" that allow the programmer to define the synchronization that occurs among concurrent procedures or tasks.

Two important issues in concurrency control are known as deadlocks and race conditions. Deadlock occurs when a resource held indefinitely by one process is requested by two or more other processes simultaneously. As a result, none of the processes that call for the resource can continue; they are deadlocked, waiting for the resource to be freed. An operating system can handle this situation with various prevention or detection and recovery techniques. A race condition, on the other hand, occurs when two or more concurrent processes assign a different value to a variable, and the result depends on which process assigns the variable first (or last).

Preventing deadlocks and race conditions is fundamentally important, since it ensures the integrity of the underlying application. A general prevention strategy is called process synchronization. Synchronization requires that one process wait for another to complete some operation before proceeding. For example, one process (a writer) may be writing data to a certain main memory area, while another process (a reader) may want to read data from that area. The reader and writer must be synchronized so that the writer does not overwrite existing data until the reader has processed it. Similarly, the reader should not start to read until data has been written in the area.

With the advent of networks, distributed computing became feasible. A distributed computation is one that is carried out by a group of linked computers working cooperatively. Such computing usually requires a distributed operating system to manage the distributed resources. Important concerns are workload sharing, which attempts to take advantage of access to multiple computers to complete jobs faster; task migration, which supports workload sharing by efficiently distributing jobs among machines; and automatic task replication, which occurs at different sites for greater reliability.

## Platform-based development

Platform-based development is concerned with the design and development of applications for specific types of computers and operating systems ("platforms"). Platform-based development takes into account system-specific characteristics,

such as those found in Web programming, multimedia development, mobile application development, and robotics. Platforms such as the Internet or an Android tablet enable students to learn within and about environments constrained by specific hardware, application programming interfaces (APIs), and special services. These environments are sufficiently different from "general purpose" programming to warrant separate research and development efforts.

For example, consider the development of an application for an Android tablet. The Android programming platform is called the Dalvic Virtual Machine (DVM), and the language is a variant of Java. However, an Android application is defined not just as a collection of objects and methods but, moreover, as a collection of "intents" and "activities," which correspond roughly to the GUI screens that the user sees when operating the application. XML programming is needed as well, since it is the language that defines the layout of the application's user interface. Finally, I/O synchronization in Android application development is more demanding than that found on conventional platforms, though some principles of Java file management carry over.

Real-time systems provide a broader setting in which platform-based development takes place. The term *real-time systems* refers to computers embedded into cars, aircraft, manufacturing assembly lines, and other devices to control processes in real time. Frequently, real-time tasks repeat at fixed-time intervals. For example, sensor data are gathered every second, and a control signal is generated. In such cases, scheduling theory is used to determine how the tasks should be scheduled on a given processor. A good example of a system that requires real-time action is the antilock braking system (ABS) on an automobile; because it is critical that the ABS instantly reacts to brake-pedal pressure and begins a program of pumping the brakes, such an application is said to have a hard deadline. Other real-time systems are said to have soft deadlines, in that no disaster will happen if the system's response is slightly delayed; an example is an order shipping and tracking system. The concept of "best effort" arises in real-time system design, because soft deadlines sometimes slip and hard deadlines are sometimes met by computing a less than optimal result. For example, most details on an air traffic controller's screen are approximations (e.g., altitude) that need not be computed more precisely (e.g., to the nearest inch) in order to be effective.

## Programming languages

Programming languages are the languages with which a programmer implements a piece of software to run on a computer. The earliest programming languages were assembly languages, not far removed from the binary-encoded instructions directly executed by the computer. By the mid-1950s, programmers began to use higher-level languages.

Two of the first higher-level languages were FORTRAN (Formula Translator) and ALGOL (Algorithmic Language), which allowed programmers to write algebraic expressions and solve scientific computing problems. As learning to program became increasingly important in the 1960s, a stripped-down version of FORTRAN called BASIC (Beginner's All-Purpose Symbolic Instruction Code) was developed at Dartmouth College. BASIC quickly spread to other academic institutions, and by 1980 versions of BASIC for personal computers allowed even students at

elementary schools to learn the fundamentals of programming. Also, in the mid-1950s, COBOL (Common Business-Oriented Language) was developed to support business programming applications that involved managing information stored in records and files.

The trend since then has been toward developing increasingly abstract languages, allowing the programmer to communicate with the machine at a level ever more remote from machine code. COBOL, FORTRAN, and their descendants (Pascal and C, for example) are known as imperative languages, since they specify as a sequence of explicit commands how the machine is to go about solving the problem at hand. These languages were also known as procedural languages, since they allowed programmers to develop and reuse procedures, subroutines, and functions to avoid reinventing basic tasks for every new application.

Other high-level languages are called functional languages, in that a program is viewed as a collection of (mathematical) functions and its semantics are very precisely defined. The best-known functional language of this type is LISP (List Processing), which in the 1960s was the mainstay programming language for AI applications. Successors to LISP in the AI community include Scheme, Prolog, and C and C++ (*see below*). Scheme is similar to LISP except that it has a more formal mathematical definition. Prolog has been used largely for logic programming, and its applications include natural language understanding and expert systems such as MYCIN. Prolog is notably a so-called nonprocedural, or declarative, language in the sense that the programmer specifies what goals are to be accomplished but not how specific methods are to be applied to attain those goals. C and C++ have been used widely in robotics, an important application of AI research. An extension of logic programming is constraint logic programming, in which pattern matching is replaced by the more general operation of constraint satisfaction.

Another important development in programming languages through the 1980s was the addition of support for data encapsulation, which gave rise to object-oriented languages. The original object-oriented language was called Smalltalk, in which all programs were represented as collections of objects communicating with each other via message-passing. An object is a set of data together with the methods (functions) that can transform that data. Encapsulation refers to the fact that an object's data can be accessed only through these methods. Object-oriented programming has been very influential in computing. Languages for object-oriented programming include C++, Visual BASIC, and Java.

Java is unusual because its applications are translated not into a particular machine language but into an intermediate language called Java Bytecode, which runs on the Java Virtual Machine (JVM). Programs on the JVM can be executed on most contemporary computer platforms, including Intel-based systems, Apple Macintoshes, and various Android-based smartphones and tablets. Thus, Linux, iOS, Windows, and other operating systems can run Java programs, which makes Java ideal for creating distributed and Web-based applications. Residing on Web-based servers, Java programs may be downloaded and run in any standard Web browser to provide access to various services, such as a client interface to a game or entry to a database residing on a server.

At a still higher level of abstraction lie declarative and scripting languages, which are strictly interpreted languages and often drive applications running in Web

browsers and mobile devices. Some declarative languages allow programmers to conveniently access and retrieve information from a database using "queries," which are declarations of what to do (rather than how to do it). A widely used database query language is SQL (Structured Query Language) and its variants (e.g., MySQL and SQLite). Associated with these declarative languages are those that describe the layout of a Web page on the user's screen. For example, HTML (HyperText Markup Language) supports the design of Web pages by specifying their structure and content. Gluing the Web page together with the database is the task of a scripting language (e.g., PHP), which is a vehicle for programmers to integrate declarative statements of HTML and MySQL with imperative actions that are required to effect an interaction between the user and the database. An example is an online book order with Amazon.com, where the user queries the database to find out what books are available and then initiates an order by pressing buttons and filling appropriate text areas with his or her ordering information. The software that underlies this activity includes HTML to describe the content of the Web page, MySQL to access the database according to the user's requests, and PHP to control the overall flow of the transaction.

Computer programs written in any language other than machine language must be either interpreted or translated into machine language ("compiled"). As suggested above, an interpreter is software that examines a computer program one instruction at a time and calls on code to execute the machine operations required by that instruction.

A compiler is software that translates an entire computer program into machine code that is saved for subsequent execution whenever desired. Much work has been done on making both the compilation process and the compiled code as efficient as possible. When a new language is developed, it is usually interpreted at first. If it later becomes popular, a compiler is developed for it, since compilation is more efficient than interpretation.

There is an intermediate approach, which is to compile code not into machine language but into an intermediate language (called a virtual machine) that is close enough to machine language that it is efficient to interpret, though not so close that it is tied to the machine language of a particular computer. It is this approach that provides the Java language with its computer platform independence via the JVM.

## Security and information assurance


**A trojan is a type of malware Screenshot of Beast, a Remote Administration Tool, or trojan, software program.**

Security and information assurance refers to policy and technical elements that protect information systems by ensuring their availability, integrity, authentication, and appropriate levels of confidentiality. Information security concepts occur in many areas of computer science, including operating systems, computer networks, databases, and software.

Operating system security involves protection from outside attacks by malicious software that interferes with the system's completion of ordinary tasks. Network security provides protection of entire networks from attacks by outsiders. Information in databases is especially vulnerable to being stolen, destroyed, or modified maliciously when the database server is accessible to multiple users over a network. The first line of defense is to allow access to a computer only to authorized users by authenticating those users by a password or similar mechanism.

However, clever programmers (known as hackers) have learned how to evade such mechanisms by designing computer viruses, programs that replicate themselves, spread among the computers in a network, and "infect" systems by destroying resident files and applications. Data can be stolen by using devices such as "Trojan horses," programs that carry out a useful task but also contain hidden malicious code, or simply by eavesdropping on network communications. The need to protect sensitive data (e.g., to protect national security or individual privacy) has led to advances in cryptography and the development of encryption standards that provide a high level of confidence that the data is safe from decoding by even the most clever attacks.

## Software engineering

Software engineering is the discipline concerned with the application of theory, knowledge, and practice to building reliable software systems that satisfy the computing requirements of customers and users. It is applicable to small-, medium-, and large-scale computing systems and organizations. Software engineering uses engineering methods, processes, techniques, and measurements. Software development, whether done by an individual or a team, requires choosing the most appropriate tools, methods, and approaches for a given environment.

Software is becoming an ever larger part of the computer system and has become complicated to develop, often requiring teams of programmers and years of effort. Thus, the development of a large piece of software can be viewed as an engineering task to be approached with care and attention to cost, reliability, and maintainability of the final product. The software engineering process is usually described as consisting of several phases, called a life cycle, variously defined but generally consisting of requirements development, analysis and specification, design, construction, validation, deployment, operation, and maintenance.

Concern over the high failure rate of software projects has led to the development of nontraditional software development processes. Notable among these is the agile software process, which includes rapid development and involves the client as an active and critical member of the team. Agile development has been effectively used in the development of open-source software, which is different from proprietary software because users are free to download and modify it to fit their particular

application needs. Particularly successful open-source software products include the Linux operating system, the Firefox Web browser, and the Apache OpenOffice word processing/spreadsheet/presentation suite.

Regardless of the development methodology chosen, the software development process is expensive and time-consuming. Since the early 1980s, increasingly sophisticated tools have been built to aid the software developer and to automate the development process as much as possible. Such computer-aided software engineering (CASE) tools span a wide range of types, from those that carry out the task of routine coding when given an appropriately detailed design in some specified language to those that incorporate an expert system to enforce design rules and eliminate software defects prior to the coding phase.

As the size and complexity of software has grown, the concept of reuse has become increasingly important in software engineering, since it is clear that extensive new software cannot be created cheaply and rapidly without incorporating existing program modules (subroutines, or pieces of computer code). One of the attractive aspects of object-oriented programming is that code written in terms of objects is readily reused. As with other aspects of computer systems, reliability (usually rather vaguely defined as the likelihood of a system to operate correctly over a reasonably long period of time) is a key goal of the finished software product.

Sophisticated techniques for testing software have also been designed. For example, unit testing is a strategy for testing every individual module of a software product independently before the modules are combined into a whole and tested using "integration testing" techniques.

The need for better-trained software engineers has led to the development of educational programs in which software engineering is a separate major. The recommendation that software engineers, similar to other engineers, be licensed or certified has gained increasing support, as has the process of accreditation for software engineering degree programs.

## Social and professional issues

Computer scientists must understand the relevant social, ethical, and professional issues that surround their activities. The ACM Code of Ethics and Professional Conduct provides a basis for personal responsibility and professional conduct for computer scientists who are engaged in system development that directly affects the general public.

As the computer industry has developed increasingly powerful processors at lower costs, microprocessors have become ubiquitous. They are used to control automated assembly lines, traffic signal systems, and retail inventory systems and are embedded in consumer products such as automobile fuel-injection systems, kitchen appliances, audio systems, cell phones, and electronic games.

Computers and networks are everywhere in the workplace. Word and document processing, electronic mail, and office automation are integrated with desktop computers, printers, database systems, and other tools using wireless networks and widespread Internet access. Such changes ultimately make office work much more efficient, though not without cost for purchasing and frequently upgrading the

necessary hardware and software as well as for training workers to use the new technology.

Computer-integrated manufacturing (CIM) is a technology arising from the application of computer science to manufacturing. The technology of CIM emphasizes that all aspects of manufacturing should be not only computerized as much as possible but also linked together via a network. For example, the design engineer's workstation should be linked into the overall system so that design specifications and manufacturing instructions may be sent automatically to the shop floor. The inventory databases should be connected as well, so product inventories may be incremented automatically and supply inventories decremented as manufacturing proceeds. An automated inspection system (or a manual inspection station supplied with online terminal entry) should be linked to a quality-control system that maintains a database of quality information and alerts the manager if quality is deteriorating and possibly even provides a diagnosis as to the source of any problems that arise. Automatically tracking the flow of products from station to station on the factory floor allows an analysis program to identify bottlenecks and recommend replacement of faulty equipment.

For example, computer technology has been incorporated into automobile design and manufacturing. Computers are involved (as CAD systems) not only in the design of cars but also in the manufacturing and testing process. Modern automobiles include numerous computer chips that analyze sensor data and alert the driver to actual and potential malfunctions. Although increased reliability has been achieved by implementing such computerization, a drawback is that only automotive repair shops with a large investment in high-tech diagnostic tools for these computerized systems can handle any but the simplest repairs.

The rapid growth of smartphones has revolutionized the telephone industry. Individuals often abandoned their landlines in favour of going completely mobile; the reluctance to pay twice for telephone service was the major driver in this decision. The telephone system itself is simply a multilevel computer network that includes radio wave links and satellite transmission, along with software switches to route calls to their destinations. If one node through which a cross-country call would normally be routed is very busy, an alternative routing can be substituted. A disadvantage is the potential for dramatic and widespread failures; for example, a poorly designed routing and flow-control protocol can cause calls to cycle indefinitely among nodes without reaching their destinations unless a system administrator intervenes.

Banking and commerce have been revolutionized by computer technology. Thanks to the Internet, individuals and organizations can interact with their bank accounts online, performing fund transfers and issuing checks from the comfort of their homes or offices. Deposits and withdrawals are instantly logged into a customer's account, which is stored on a remote server. Computer-generated monthly statements are unlikely to contain errors. Credit and debit card purchases are also supported by computer networks, allowing the amount of a transaction to be immediately deducted from the customer's account and transferred to the seller's. Similarly, networks allows individuals to obtain cash instantly and almost worldwide by stepping up to an automated teller machine (ATM) and providing the proper card and personal identification number (PIN).

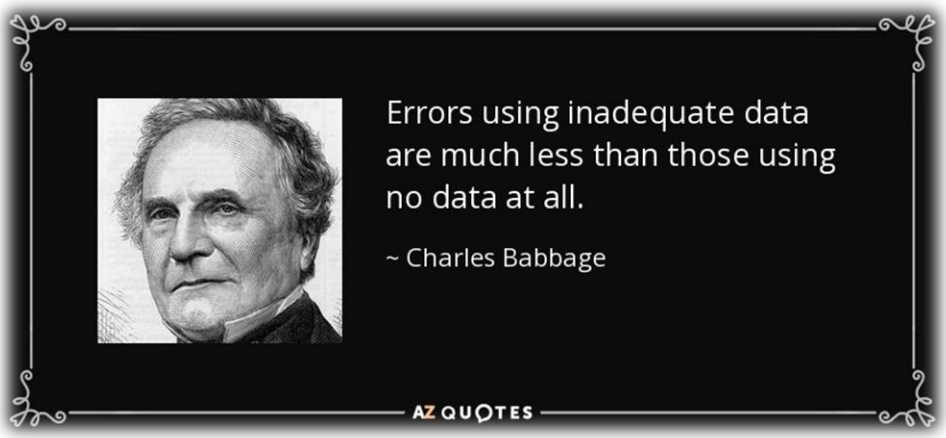
The security challenges associated with these technologies are significant. Intruders can intercept packets traveling on a network (e.g., being transported via a satellite link) and can decrypt them to obtain confidential information on financial transactions. Network access to personal accounts has the potential to let intruders not only see how much money an individual has but also transfer some of it elsewhere. Fortunately, increased software security measures have made such intrusions less likely.

Computer technology has had a significant impact on the retail industry. All but the smallest shops in places with Internet access have replaced the old-fashioned cash register with a terminal linked to a computer system. Some terminals require that the clerk type in the code for the item, but most checkout counters include a bar-code scanner, which reads into the computer the Universal Product Code (UPC) printed on each package. Cash register receipts then include brief descriptions of the items purchased (by fetching them from the computer database), and the purchase information is also relayed back to the computer to cause an immediate adjustment in the inventory data. The inventory system can easily alert the manager when the supply of an item drops below a specified threshold. In the case of retail chains linked by networks, the order for a new supply of an item may be automatically generated and sent electronically to the supply warehouse. In a less extensively automated arrangement, the manager can send in the order electronically by a direct link to the supplier's computer. These developments have made shopping much more convenient. The checkout process is faster, checkout lines are shorter, and desired items are more likely to be in stock. In addition, cash register receipts contain more detailed information than a simple list of item prices; for example, many receipts include discount coupons based on the specific items purchased by the shopper.

Since the mid-1990s one of the most rapidly growing retail sectors has been electronic commerce, involving use of the Internet and proprietary networks to facilitate business-to-business (B2B), consumer, and auction sales of everything imaginable—from computers and electronics to books, recordings, automobiles, and real estate. Popular sites for electronic commerce include Amazon, eBay, and the websites for most large retail chain stores.

@@@@@@@@@@@@@@

# QUOTES



## Kindly visit these Web Links for more Quotes:

[01] https://mathshistory.st-andrews.ac.uk/Biographies/Babbage/quotations/

[02] https://www.brainyquote.com/authors/charles-babbage-quotes

[03] https://todayinsci.com/B/Babbage_Charles/BabbageCharles-Quotations.htm

[04] https://quotefancy.com/charles-babbage-quotes

[05] https://in.pinterest.com/pin/979955200144078782/

[06] https://www.azquotes.com/author/725-Charles_Babbage

# History of Computers

# Timeline

https://www.superprof.co.in/blog/the-history-of-the-computer/

The word "computer" comes from the Latin "*computare*, "to calculate". A computer was originally a mechanical calculator or a person who did calculations where they were needed (such as engineering companies), first by hand, then with the help of a mechanical calculator.

Much of the math needed for the early space programs was done by human computers, most of them women.

So how, from the elaboration of true mathematics in Antiquity to the first commercial computer sold in 1951 (the UNIAC), to IBM's first Personal Computer (PC) in 1981 to the graphic interface of Windows 95, did we get where we are today?

The world has transitioned from the Industrial Age to the Information Age. But Rome wasn't built in a day: modern-day computing was established over many decades with the help of dozens of mathematicians, physicians, and theorists.

**Timeline Of Computer: From Early Computer to Modern Machines**

**3000 BC**

**Abacus (3000 BC)**

Considered one of the earliest computing devices, the abacus was a manual counting tool used for arithmetic calculations.

**17th Century**

**Mechanical Calculators (17th century)**

Inventors like Blaise Pascal and Gottfried Wilhelm Leibniz created mechanical calculators to perform basic mathematical operations.

**1837**

**Analytical Engine (1837)**

Designed by Charles Babbage, the Analytical Engine was an early concept of a general-purpose computer. Although it was never built, it laid the foundation for modern computing.

**19th Century**

**Tabulating Machines (late 19th century)**

Herman Hollerith developed machines that used punched cards to process and tabulate data. These machines were widely used for census calculations.

**1930s - 1940s**

**Vacuum Tube Computers (1930s-1940s)**

Electronic computers using vacuum tubes as basic components were developed, such as the Atanasoff-Berry Computer (ABC) and the Colossus, used for code-breaking during World War II.

**1946**

**ENIAC**

ENIAC (Electronic Numerical Integrator and Computer) was one of the earliest general-purpose electronic computers. It used vacuum tubes and was capable of performing complex calculations.

**1947**

**Transistors (1947)**

The invention of transistors by John Bardeen, Walter Brattain, and William Shockley revolutionized computer technology, making computers smaller, faster, and more reliable.

**1958**

**Integrated Circuits (1958)**

Jack Kilby and Robert Noyce independently developed integrated circuits, which combined multiple electronic components on a single chip. This led to the miniaturization and increased efficiency of computers.

**1971**

**Microprocessors (1971)**

Intel introduced the first microprocessor, the Intel 4004, which integrated all the central processing unit (CPU) functions onto a single chip. This advancement paved the way for personal computers.

**1970s - 1980's**

**Personal Computers**

Companies like Apple, IBM, and Microsoft introduced affordable personal computers that were user-friendly and widely adopted by individuals and businesses.

### 1980s

**Graphical User Interfaces (1980s)**

The introduction of graphical user interfaces (GUIs) made computers more accessible to non-technical users. Interfaces like Apple's Macintosh and Microsoft's Windows revolutionized computer interaction.

### 1990s

**Birth of Internet**

The development of the World Wide Web and the widespread adoption of the Internet transformed computers into powerful communication and information-sharing tools.

### 2000s

**Laptops, Tablets, and Smartphones (2000s)**

The advancement of technology led to the development of portable computing devices like laptops, tablets, and smartphones, making computing more mobile and accessible.

### 2000s - 2010s

**Cloud Computing and Artificial Intelligence (2000s-2010s)**

The emergence of cloud computing allowed users to store and access data remotely, while advancements in artificial intelligence and machine learning enabled computers to perform complex tasks and learn from data.

### Modern Computers (Present)

### Quantum Computing (present)

Ongoing research in quantum computing aims to develop computers that harness the principles of quantum mechanics, potentially revolutionizing computing power and solving complex problems.

### Five Generations of Computers

The first generation of computers was:

| | | |
|---|---|---|
| First generation | 1940s-1950s | Vacuum tube based |
| Generations of computers | Generations timeline | Evolving hardware |
| Second generation | 1950s-1960s | Transistor based |
| Third generation | 1960s-1970s | Integrated circuit based |
| Fourth generation | 1970s-present | Microprocessor based |
| Fifth generation | The present and the future | Artificial intelligence based |

***Did you know?*** The generation of computers characterized by the use of microchips is the third generation of computers. The third generation of computers used integrated circuits, which were made up of multiple transistors and other components on a single chip of silicon.

These microchips allowed for faster processing speeds, greater storage capacity, and smaller computer sizes. The third generation of computers also marked the beginning of the use of high-level programming languages such as COBOL and FORTRAN.

# Evolution of Computers from Algorithms to the First Programme

## Al-Khwarizmi, the father of Algebra and algorithms

**al-Khwarizmi, the father of algorithms, without which there would be no computer programs.**

Let's start with Abu Jaffar al-Khwarizmi, also called Mr. Algorithm - the word "algorithm" is, in fact, a latinisation of his name.

The development of computers is actually very closely linked to fundamental research in mathematics, most particularly logic and the algorithms that al-Khwarizmi elaborated in the 9th century AD. We also owe our modern Arabic numerals (1,5, 10 as opposed to the the Latin I, V, X) to him.

To calculate is basically to resolve a specific problem while using a specific set of rules.

Algebra - and algorithms - is the science of organising the operations needed to complete a task. These are abstract operations - for example, adding two numbers together. To go from there to a concrete operation, you have to code the abstract ideoa into a specific language, such as single signs to represent a value, with the placement of these signs influencing their information - "1" has a different meaning in the number "31" and "13" - and various other symbols encoding certain operations, such as "+", "-" and "=" (all of which, however, came later than al-Kharizmi.)

This is how we go from "I have two apples and my friend gives me two more" to 2 + 2 = ?

al-Khwarizmi gave mathematics its own "programming" language.

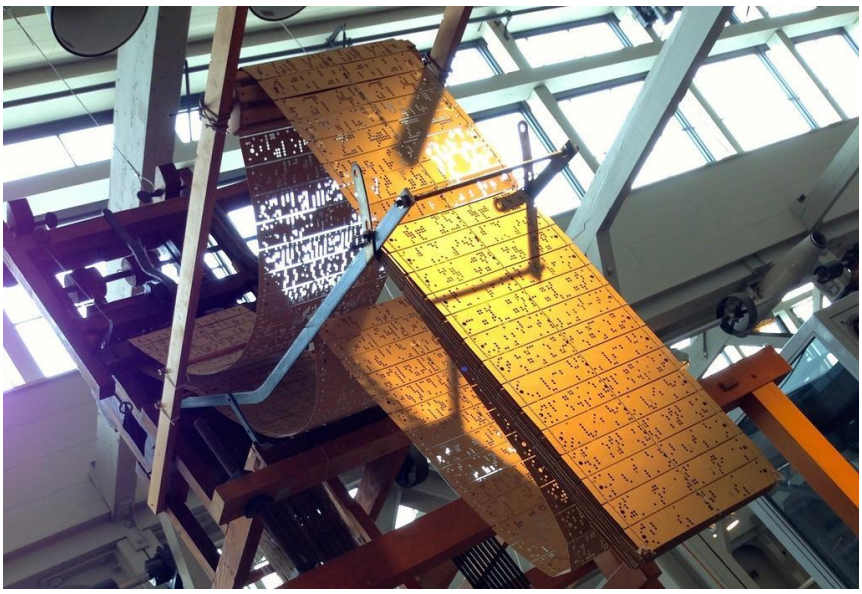
Improve your coding skills by taking programming courses on Superprof.

## The First Computer Programme

The first programme to influence the running of a man-made system was not for a computing machine, but for a mechanical loom. In 1801, the famous French weaver Joseph Marie Jacquard introduced a mechanical loom that could be programmed for different motifs using punch cards created using special typewriter-like machines.

The position of the punches changed the position of the loom's **mechanical parts**, chose which shuttles to use, etc. - rather like modern industrial robots. One punch card equaled one row of the motif being woven; the cards were bound together to make strips containing a whole piece - rug, wall hanging, upholstery fabric etc.

Check here for creditable **IT security courses** now.



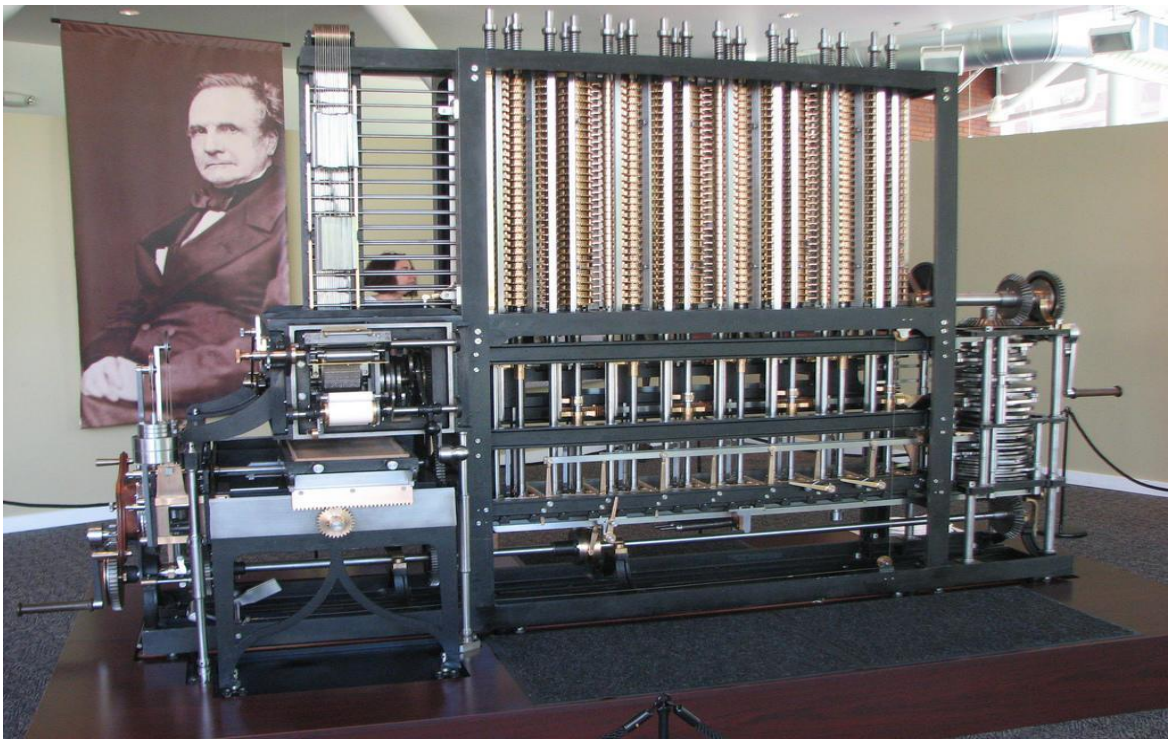
**A program for a Jaquard loom - a long roll of punch cards coding a specific design. Photo credit: rumpleteaser on Visual hunt / CC BY**

Charles Babbage planned to used punch cards for his Analytical Engine, and the Harvard Mark I later used punched rolls of paper for programming.

## Ada Lovelace and the Analytical Engine

In the end, though, computing aims to remove thought from calculations in order to make it possible for a machine - incredibly fast but entirely without thought - to calculate by itself.

Charles Babbage is considered the father of modern computers. He was never able to finish his Difference Engine (a machine that calculates polynomial functions) as the research and testing took so long that the Crown cut his funding. However, his son eventually finished building the first machine. It can be seen at the London Science Museum and it still works!



**Charles Babbage's Difference Engine was never completed in his lifetime - but subsequent construction prove that it worked.**

He spent the rest of his life on the more complicated Analytical Engine, which had an arithmetic logic unit.

Ada Lovelace, a 19th-century mathematician, first published her work in 1840 under a masculine name. She is credited with writing the first computer programme for Charles Babbage's Analytical Engine. The Analytical Engine was supposed to execute any calculation demanded of it by Man: both symbolical and numerical operations.

## Kindly visit these Web Links for MORE info:

https://www.hp.com/ca-en/shop/offer.aspx?p=a-timeline-of-computer-history-and-development

https://www.livescience.com/20718-computer-history.html

https://www.computerhistory.org/timeline/computers/

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://history.computer.org/pubs/timeline.pdf

https://www.computinghistory.org.uk/cgi/computing-timeline.pl

(😊)@@@@@@@@@@@(😊)

https://www.computer.org/volunteering/awards/babbage

# IEEE CS Charles Babbage Award

In Recognition of Significant Contributions in the Field of Parallel Computing.

Established in memory of **Charles Babbage** in recognition of significant contributions in the field of parallel computation. The candidate would have made an outstanding, innovative contribution or contributions to parallel computation. It is hoped, but not required, that the winner will have also contributed to the parallel computation community through teaching, mentoring, or community service.

This award covers all aspects of parallel computing including computational aspects, novel applications, parallel algorithms, theory of parallel computation, parallel computing technologies, among others.

A certificate and a $1,000 honorarium presented to a single recipient. The winner will be invited to present a paper and/or presentation at the annual IEEE CS International Parallel and Distributed Processing Symposium (IPDPS).

The IPDPS conference originated in 1986 as an activity of the Orange County (California) Chapter of the IEEE Computer Society Chapter. The founders of the first International Parallel Processing Symposium (IPPS) were primarily engineers from the semiconductor field and academics interested in computer architecture and high-performance processing, looking to solve problems of computation through architectural and software innovation.

Past Recipients

2025

Srinivas Aluru

For pioneering contributions to the field of Parallel Computational Biology.

2024

Franck Cappello

For pioneering contributions and inspiring leadership in distributed computing, high-performance computing, resilience, and data reduction.

2023

Keshav Pingali

For contributions to high-performance compilers and graph computing.

2022

Dhabaleswar K. (DK) Panda

For contributions to high performance and scalable communication in parallel and high-end computing systems..

2021

Guy Blelloch

For contributions to parallel programming, parallel algorithms, and the interface between them.

2020

Yves Robert

For contributions to parallel algorithms and scheduling techniques.

2019

Ian T. Foster

For sustained contributions to high-performance computing and distributed systems at the highest level.

2017

Mateo Valero

For contributions to parallel computation through brilliant technical work, mentoring PhD students, and building an incredibly productive European research environment.

2015

Alan Edelman

2014

Peter Kogge

2013

James Demmel

2012

Chris Johnson

2011

Jack Dongarra

2010

Burton Smith

2009

Wen-Mei Hwu

2008

Joel Saltz

2007

Mike Flynn

2006

Bill Dally

2005

Yale N. Patt

2004

Christos Papadimitriou

2003

Michel Cosnard

2002

Steve Wallach

2001

Thomson Leighton

2000

Michael O. Rabin

1999

K. Mani Chandy

1998

Jim Gray

1997

Frances Allen

1995

Richard Karp

1994

Arvind

1993

K. Mani Chandy

1992

David Kuck

1991

Harold Stone

1990

H.T. Kung

1989

Irving S. Reed

(😉)@@@@@@@@@@@@@@@@(😉)

# Memorials


**Green plaque in London**

There is a black plaque commemorating the 40 years Babbage spent at 1 Dorset Street, London. Locations, institutions and other things named after Babbage include:

- The Moon crater, Babbage
- The Charles Babbage Institute, an information technology archive and research center at the University of Minnesota
- Babbage River Falls, Yukon, Canada
- The Charles Babbage Premium, an annual computing award
- British Rail named a locomotive after Charles Babbage in the 1990s.
- Babbage Island, Western Australia
- The Babbage Building at the University of Plymouth, where the university's school of computing is based
- The Babbage programming language for GEC 4000 series minicomputers
- "Babbage", *The Economist*'s Science and Technology blog
- The former chain retail computer and video-games store "Babbage's" (now GameStop) was named after him.

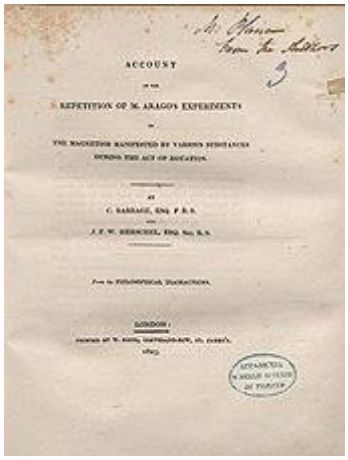
## In fiction and film

Babbage frequently appears in steampunk works; he has been called an iconic figure of the genre. Other works in which Babbage appears include:

- The 2008 short film *Babbage*, screened at the 2008 Cannes Film Festival, a 2009 finalist with Haydenfilms, and shown at the 2009 HollyShorts Film Festival and other international film festivals. The film shows Babbage at a dinner party, with guests discussing his life and work.
- Sydney Padua created *The Thrilling Adventures of Lovelace and Babbage*, a cartoon alternate history in which Babbage and Lovelace succeed in building the Analytical Engine. It quotes heavily from the writings of Lovelace, Babbage and their contemporaries.

- Kate Beaton, cartoonist of webcomic *Hark! A Vagrant*, devoted one of her comic strips to Charles and Georgiana Babbage.
- The *Doctor Who* episode "Spyfall, Part 2" (Season 12, episode 2) features Charles Babbage and Ada Gordon as characters who assist the Doctor when she's stuck in the year 1834.

## Publications



ACCOUNT

OF THE

REPETITION OF M. ARAGO'S EXPERIMENTS

ON

THE MAGNETISM MANIFESTED BY VARIOUS SUBSTANCES

DURING THE ACT OF ROTATION.

BY

C. BABBAGE, ESQ. F.R.S.

AND

J.F.W. HERSCHEL, ESQ. M.A. F.R.S.

LONDON:

***Account of the repetition of M. Arago's experiments on the magnetism manifested by various substances during the act of rotation*, 1825**

- *Account of the repetition of M. Arago's experiments on the magnetism manifested by various substances during the act of rotation. London: William Nicol. 1825.*
- *Babbage, Charles (1826). A Comparative View of the Various Institutions for the Assurance of Lives. London: J. Mawman. charles babbage.*
- *Babbage, Charles (1830). Reflections on the Decline of Science in England, and on Some of Its Causes. London: B. Fellowes. charles babbage.*
- *Abstract of a paper entitled Observations on the Temple of Serapis at Pozzuoli. London: Richard Taylor. 1834.*
- *Babbage, Charles (1835). On the Economy of Machinery and Manufactures (4th ed.). London: Charles Knight.*
- *Babbage, Charles (1837). The Ninth Bridgewater Treatise, a Fragment. London: John Murray. charles babbage.* (Reissued by Cambridge University Press 2009, ISBN 978-1-108-00000-0.)
- *Babbage, Charles (1841). Table of the Logarithms of the Natural Numbers from 1 to 108000. London: William Clowes and Sons. charles babbage.* (The LOCOMAT site contains a reconstruction of this table.)
- *Babbage, Charles (1851). The Exposition of 1851. London: John Murray. charles babbage.*
- *Laws of mechanical notation. 1851.*
- *Babbage, Charles (1864). Passages from the Life of a Philosopher. London: Longman.*
- *Babbage, Charles (1989). Hyman, Anthony (ed.). Science and Reform: Selected Works of Charles Babbage. Cambridge University Press. ISBN 978-0-521-34311-4.*
- *Babbage, Charles (1989) [1815]. Charles Babbage's Lectures On Astronomy. London.*

@@@@@@@@@@@@@@

# The Father of the Computer

https://www.elephantlearning.com/post/charles-babbage-the-father-of-the-computer

***By Ashley Langham***

*********

Without Charles Babbage we wouldn't be in the digital era, as we are today. Charles Babbage was the inventor of the original mechanical computer, called the Difference Engine, which has led subsequent inventors to create more complex electronic designs. In 1823, he even went on to win the gold medal for his invention, from the Royal Astronomical Society, in the UK, for which he founded in 1820.

Babbage was not a very happy man throughout his adult life. Amongst his woes were sudden personal tragedies; and he felt the government didn't properly support him and adequately fund his inventions, making it hard for him to fully realize the improvements he wanted to make, to the Difference Engine.

Still, the Difference Engine made for impressive technology, despite it not meeting his own vision. Plus, Babbage created other inventions that improved the way of life for his time, and for generations to come. Read on to learn more about his work and life!

Charles Babbage was born in London to banker Benjamin Babbage. He was a sickly child so he had to be homeschooled with private tutors for most of his childhood; but his health did improve closer to the time he went off to college. In 1810, Charles Babbage decided to study math at Trinity College of Cambridge University. He became very uninspired and unimpressed with the instructors there because he felt he was more advanced than they were. In 1812, he and some of his friends decided to create and join an anti-teaching group called Analytical Society, to explore more advanced issues in mathematics. He was genuinely disturbed by the miscalculations in logarithmic tables, he uncovered, done by some of the instructors.

Before Charles Babbage attended University, he taught himself a deep understanding of contemporary mathematics. He read mathematicians' publications from Robert Woodhouse, Joseph Louis Lagrange, and Maria Agnesi while he was a young teenager, where his love for mathematics began. (Oxford Dictionary of Scientists)

Also in 1812, he transferred to Peterhouse, Cambridge, another college at the University, where he found himself top of his class in mathematics. He wanted to improve the calculations in logarithmic tables, and create a machine that could do that work automatically. This idea would eventually lead him to developing the Difference Engine.

Charles Babbage graduated from the University of Cambridge to pursue a master's degree. Before he graduated from his master's program, he lectured at the Royal Institution on Astronomy and was elected as a Fellow of the Royal Society in 1816. After graduating, he struggled to find a permanent suitable job for himself and lived off his father's finances. His father was, in today's standards, a multimillionaire, so Babbage, and his wife were well provided for.

In 1820, Babbage founded the Royal Astronomical Society. It was here he developed his ideas around how to create a machine for computation of numbers. In 1822, he would develop the Difference Engine and discussed his machine's functionality at the Royal

Astronomical Society, that same year. Originally the machine used a scribe for copying and computing numbers, rather than printing them out automatically. The machine also was loosely based off of the Jacquard machine, a 19th century device used to weave textiles.

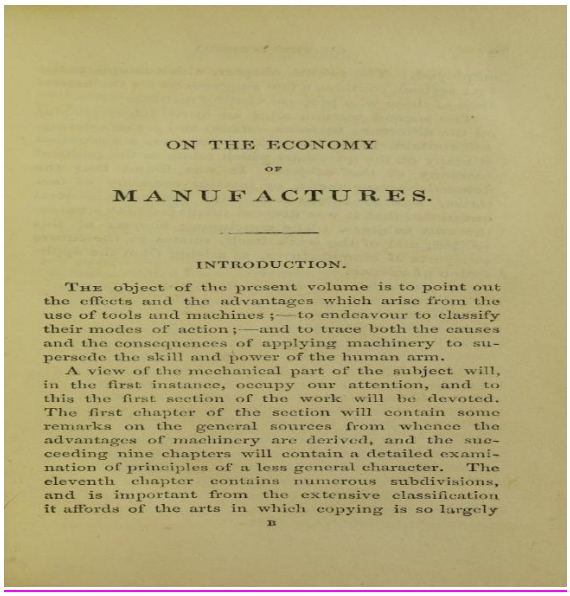
The following year, he won a prize for his invention and was promised substantial governmental funding to continue building on it, to make improvements. Babbage did not like having to answer to the UK government; they asked him many questions to understand the improvements for the machine. Babbage found their constant questioning annoying and the type of questions they asked "ridiculous". Unfortunately, neither side reached a workable relationship, and the government quickly withdrew their funding.

## Charles Babbage's Most Tragic Year

In 1827, Babbage really suffered with his personal tragedies. His father, wife, and two children died all in that year. It was suggested to him that he should take some time away from working on his inventions to heal from his grief. He decided to take a sabbatical and travel the year through continental Europe.

His father left him a substantial amount of wealth. He became a multimillionaire, in today's calculations, and had to self fund the improvements on the Difference Engine. By 1832, he completed the second prototype which was able to compile math tables automatically, but gave errors in calculations. He eventually stopped working on the machine for a while because he was so displeased with its progress and didn't have governmental funding to continue his work.

Upon his return to the UK, he received a position as the Lucasian Chair of Mathematics for Cambridge. In eleven years he held his position, he never once taught a lecture and ended up falling out of favor quickly with his colleagues. Instead, he spent most of his time writing books and focusing on political economy. In 1832, his book The Economy of Machinery and Manufacturers was published where he talked about how to make the labor market more efficient by: disseminating work in such a way that only high skilled tasks be done by highly skilled workers and menial tasks be done by lower paid workers. (Oxford Dictionary of Scientists) This idea that he came up with was eventually known as the Babbage principle.



ON THE ECONOMY
OF
MANUFACTURES.

INTRODUCTION.

The object of the present volume is to point out the effects and the advantages which arise from the use of tools and machines;—to endeavour to classify their modes of action;—and to trace both the causes and the consequences of applying machinery to supersede the skill and power of the human arm.

A view of the mechanical part of the subject will, in the first instance, occupy our attention, and to this the first section of the work will be devoted. The first chapter of the section will contain some remarks on the general sources from whence the advantages of machinery are derived, and the succeeding nine chapters will contain a detailed examination of principles of a less general character. The eleventh chapter contains numerous subdivisions, and is important from the extensive classification it affords of the arts in which copying is so largely

B

*Title page of Babbage's 'On the Economy of Machinery and Manufactures', first published in 1832*

In 1837, even without governmental funding, Charles Babbage was able to create an upgraded prototype to his Difference Engine, called the Analytical Engine. Its purpose was to perform four arithmetic operations and square root extractions. It would be this prototype that led modern inventors to develop digital computers because the programming language was analogous to the modern day assembly language.

In this prototype, punch cards were used, one for arithmetic operations, one for numerical constants, and one to transfer numbers from storage to the arithmetic unit. Unfortunately, it only ran a few tasks and had quite a few errors just doing that. Babbage became very frustrated with the machine and never fully improved it to the level he wanted to be.

On his deathbed, Babbage carried his frustration and callousness for the government with him. He felt that due to their early withdrawal of funding, he never could get the Analytical Engine to where he wanted it to be. In 1910, his son tried to continue working on his own version of the Analytical Engine, but to no avail. His version was not programmable and had no storage.

Though Babbage thought his inventions for the "computer" were not to his standard, he made great contributions in the digital era. He also invented the speedometer, occulting lights, which allowed lighthouse lights to stay on longer through periods of darkness, and the locomotive cow catcher, a device that cleared obstacles from rail tracks.

***A fun fact,***
***his brain remains on display and preserved today***
***in the Science Museum in London.***

ISBN 978-81-984404-5-7